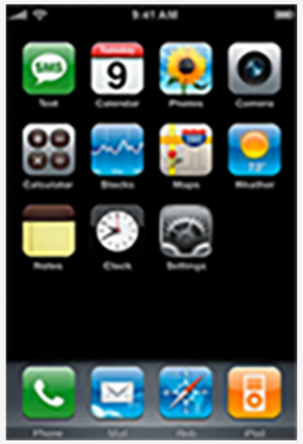
iOS开发知识精美画册

一百多张精美ppt，生动形象、切中要点地讲解iOS开发众多知识点！

互动教程网 出品

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM



iOS 1.0

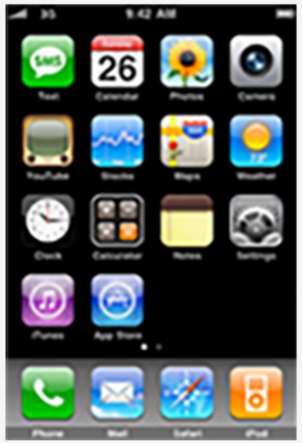
2007年

第一代iPhone OS面世时，存在着各种各样的问题，诸如无法发送MMS、全封闭的系统，并且只有黑色的背景，无法自定义壁纸等。但是这些问题并不能阻止它成为一个伟大的操作系统。

iOS 1.0拥有大量的创新功能，它展现了革命性的人机交互方式，iPhone之前的智能手机都是使用键盘或手写的方式进行人机交互，而iPhone则实现了一种全手控操作模式。所以谷歌地图在第一代iPhone上的用户体验，秒杀了当时在其他平台上的版本。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM



iOS 2.0

2008年

一年后伴随着iPhone 3G的出现，iPhone OS才算是得到了真正意义上的完整。2008年7月，Apple为iOS发布了App Store，并为第三方应用提供了一种可管理的标准模式，用来开发、浏览、下载和安装应用。

如今的App Store绝对是iOS发展史上最重要的杀手应用，它不仅帮助Apple建立了一个庞大的应用生态，而且让众多的开发者因此找到了自己的商业模式和商业机会，而Apple也因此积累了庞大数量的精品应用。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 3.0

2009年

在iPhone 3GS发布的同时，Apple也带来了iOS 3.0系统。iPhone 3GS型号中的S也代表了Speed速度的意思，由于硬件上的提升，所以3GS在速度上有了不小的升级。

从此iPhone手机每两年进入一次大升级，中间一年推出的设备型号的尾部常被标以S，如iPhone 4S、iPhone 5S等，表示在设备性能上进行了一些升级。iOS 3.0还引入了应用内购买、iTunes电影租赁、横向键盘和语音控制。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 4.0

2010年

在iPhone 4时代，iPhone OS被正式更名为iOS，iOS 4开始支持多任务技术，此时的多任务方式比较不会受到后台应用占用内存的影响，也能保证不错的续航能力。多任务的菜单是通过双击Home键的方式调出的。

iOS 4.0的其他更新还包括app文件夹、统一邮件收件箱、iBooks和iAd。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 5.0

2011年

iOS 5.0与iPhone 4S一起推出。Siri是iOS 5最大的亮点，实现了语音控制输入的功能。用户通过Siri技术，使用语音提问的方式进行人机交互。

Apple在iOS 5中取消了对数据线的依赖，可以使用Wi-Fi进行设备的激活，这意味着iPad、iPhone已经彻底变成了个人移动终端，减轻了对桌面环境的依赖。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 6.0

2012年

Siri语音助手在iOS 6中得到了加强，iOS 6用户已经可以使用语音打开应用程序，甚至可以发送状态到Facebook和Twitter。iOS 6也被称为一个告别谷歌的系统版本，最大的一个变化是不再使用从第一代开始一直内置的谷歌地图，而是采用了Apple自己开发的地图服务。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 7.0

2013年

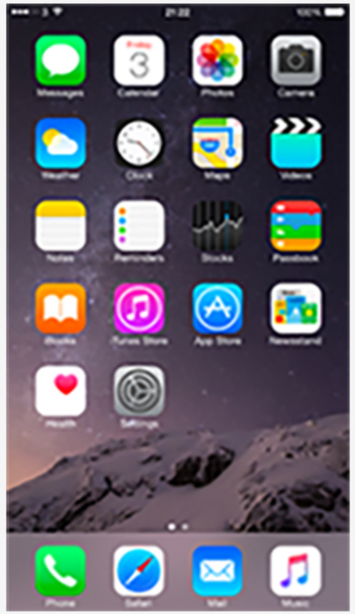
iOS系统最突出的一次演变是在iOS 7时发生的。iOS掌门人换成乔纳森后，iOS 7也进行了大调整，大家熟悉的拟物化图标全都被拍扁了，多任务界面也发生了巨大的变化，变得更加直观，用卡片式替代了原有的应用图标。

当时受Windows Phone操作系统的影响，其它两个主流移动操作系统iOS和Andriod都开始陆续向界面简洁、扁平、易操作和多彩的风格变化。

而Apple从iOS 7开始就将系统的设计风格变得扁平及鲜艳，景深切换看上去也非常炫酷，并且图标的改动也非常容易理解。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM



iOS 8.0

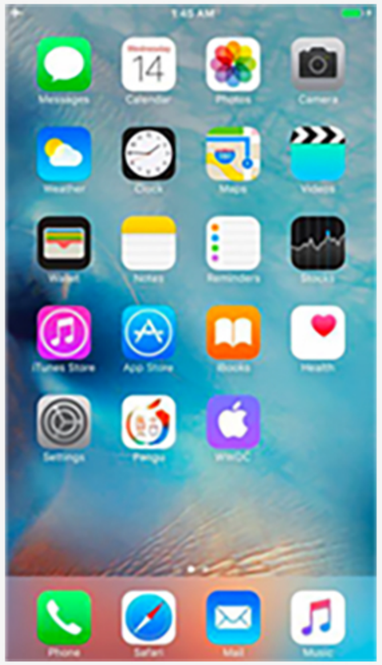
2014年

iOS 8的正式版本的系统于2014年9月17日向用户推送，它支持iPhone 4S、iPad 2、iPod touch（第五代）及更新的设备。随后8.0的更新还有Apple Pay，Apple Watch支持，iCloud Photo Library和Apple Music等。

在iOS 8中，Apple也对一直被诟病的输入法进行了升级，终于开放了对第三方输入法的支持，也正是从该版本开始，iOS越狱版的用户变得也越来越少。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM



iOS 9.0

2015年

iOS 9系统比iOS 8更加稳定，功能也更加全面，而且还更加开放。iOS 9加入了更多的新功能，包括更加智能的Siri和省电模式。Siri变得更加智能，甚至在你开口之前，新的主动建议功能就可以帮你把事情办妥。

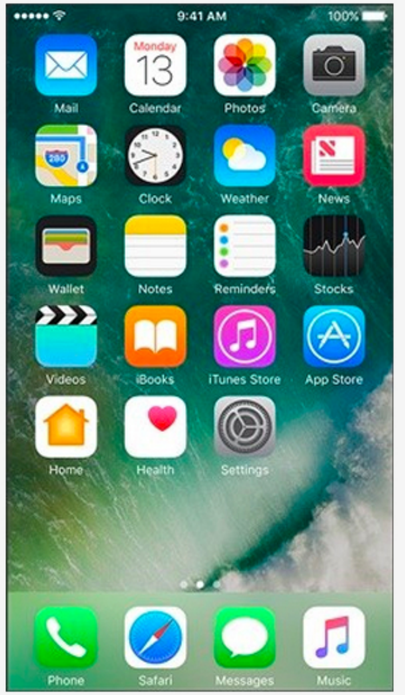
Slide Over、Split View 和画中画功可以让用户能够以全新的方式，在iPad上进行多任务处理；还有全新的QuickType功能，让快捷键触手可及，从而使格式设定和文本选择更为轻松。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 10

2016年

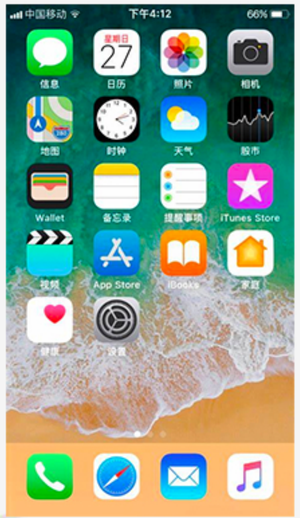


iOS 10.0涵盖了众多新特性。锁屏界面可以显示丰富的通知功能，左滑右滑可以进入不同的功能页面。Siri变得更加开放，允许开发者调用Siri API。

加入了智能脸部和场景的识别，并且增加LivePhoto编辑功能。HomeKit可以管理所有连接iOS的智能硬件，提供场景化功能。而针对中国用户，苹果对电话功能进行了十分体贴的优化，增加骚扰电话识别功能。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM



iOS 11

2017年

iOS 11 为全球最先进的手机操作系统设定了全新的标准：利用 Core ML 搭配机器学习的强大力量，使您的 app 变得更智能化；ARKit 帮助您实现美妙得仿佛身临其境的增强现实体验；更有 iPad 的拖放功能、全新的文件 app、新版相机 API、等等最新的多任务性能，助您实现更加统一和美妙的用户体验。

iOS 11 将这个先进的移动操作系统又一次提升至新标准。它不仅翻开了 iPhone 的新一页，更是开启了 iPad 的新篇章；还将游戏和 app 中的增强现实体验带到这两种设备上，让更多令人眼界大开的可能成为现实。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 12

2018年

iOS 12伴随着iPhone X而来，新增防上瘾功能，睡前免打扰、家长控制功能。勿扰模式支持睡觉时应用通知消息屏蔽、最多支持32人的FaceTime视频聊天、通知分组并且支持一键清空，App Limits应用使用时间控制，家长可以限制孩子使用设备的时长等。

全新的iOS 12最大的亮点是提升了速度和流畅性。据官方介绍，相比上一代iOS 11，iOS 12的App启动速度快了40%，输入法调出速度快了50%，相机启动速度快了70%。

iOS系统的发展历程

FEATURES OF EACH iOS SYSTEM

iOS 13

2019年

在2019苹果秋季新品发布会上，苹果宣布于2019年9月19日推送iOS 13正式版。iOS 13中，苹果将Face ID的识别速度提升了30%。除此之外，通过改进App Store应用的打包方式，App的占容大小被压缩了50%，升级占容被压缩了60%的空间。在这一轮优化后，iOS 13的app启动速度比iOS 12快了两倍之多。

深色模式（Dark Mode）是这次新增最重要的功能之一，这也是iOS 7之后苹果在iOS系统上所做的最大的一次视觉效果改变，带来系统级的深色模式，并且官方应用全部支持黑暗模式。

App Store历史沿革

HISTORY OF THE APP STORE

01

2008年7月11日

苹果APP Store正式上线。App Store中可供下载的应用已达800个，下载量达到1千万次。

02

2011年1月6日

APPStore平台上大部分应用价格低于10美元，并且有约20%的应用是免费下载的。用户购买应用所支付的费用由苹果与应用开发商三七分成。

03

2013年1月8日

App Store的应用下载量已经突破400亿次，其中半数是2012年完成的；总活跃帐户数也达5亿。

04

2017年3月

对于开发者来说，发布应用的审核周期变得更快。苹果明确告知热更新在App Store审核指南中明令禁止。

05

2018年6月7日

WWDC后苹果修改了158处审核条例，特别对用户数据收集，使用等做出严格规定，针对日前火爆的区块链项目也有重大调整。

06

2019年3月

苹果竞价广告开放46个新地区，同时下架了千余款贷款类的app，彼时金融类的贷款产品监管变得非常严格。

07

2020年1月1日

2020 年元旦一天，App Store 的用户消费额就达到 3.86 亿美元，比去年增长了 20%，创下了新的单日记录。

App Store的产业链

APP STORE'S INDUSTRIAL CHAIN

开发者

1、负责应用程序的开发；

2、自主定价、运营平台上的产品。

苹果公司

1、提供平台和开发工具包；

2、负责应用产品的营销工作；

3、负责收费并按月结算给开发者。

用户

1、应用程序的体验者；

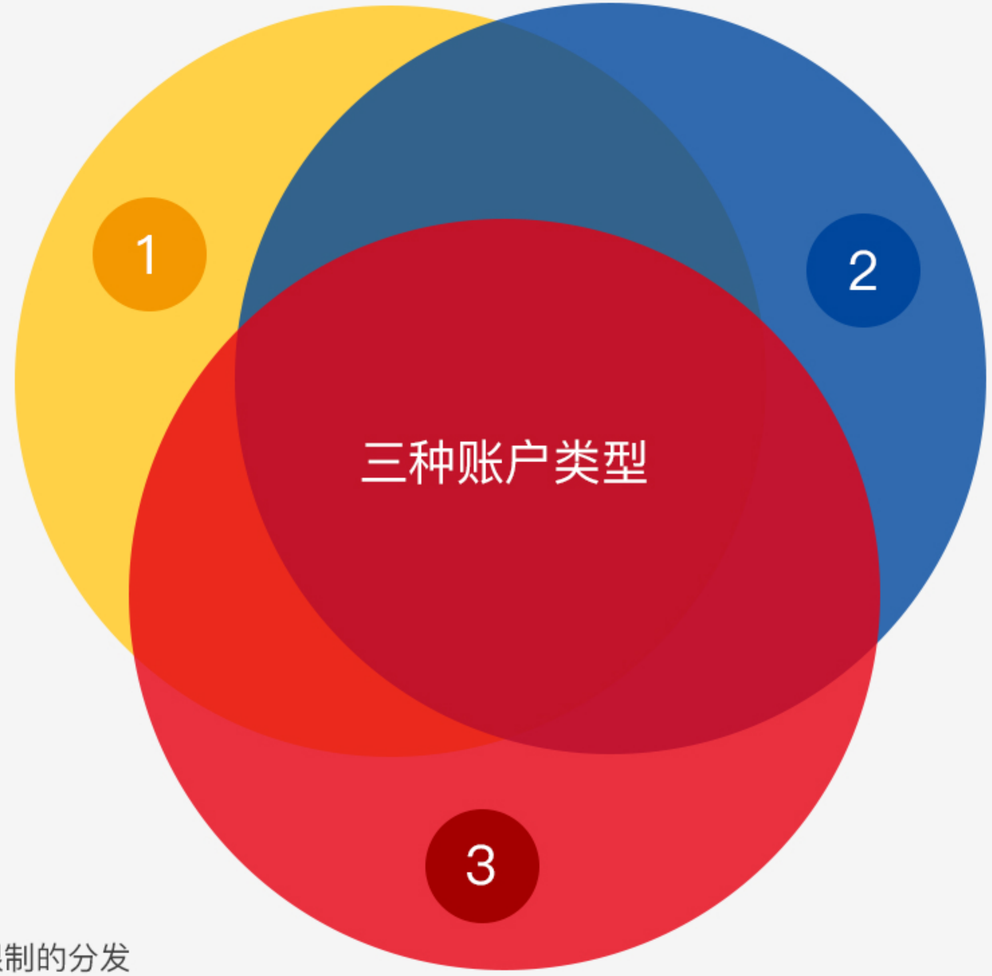
2、付费购买增值服务。

加入付费的开发者计划

JOIN THE APPLE DEVELOPER PROGRAM

个人开发者账户，

每年99美金
可以在App Store发布产品
或在100个iOS设备上进行测试
该账户在App Store中显示为个人姓名



公司开发者账户，

每年99美金
可以在App Store发布产品
或在100个iOS设备上进行测试
允许多个开发者协作开发
需要填写公司的邓白氏编码（D-U-N-S）

企业开发者账户，

每年299美金
不可以在App Store发布产品
用于在企业内部进行无设备数量限制的分发

上架App Store的五个步骤

FIVE STEPS TO THE APP STORE

审核上线

完成App的测试并打包之后
通过App Store Connect提交App
审核通过之后即可自动上线App Store
审核周期一般在两天之内

产品测试

软件测试人员相当于是APP开发的质检员
每个版本的APP完工之后
都需要交给测试人员进行功能、性能、兼容性等测试
借助TestFlight工具实现测试的流水化

开发产品

产品经理做需求分析，并且整理成需求文档
UI设计师负责App的界面设计和交互设计
iOS开发工程师实现App的界面和功能
开发者可以灵活采用Github中的开源框架
快速实现App的开发

准备工作

硬件环境：苹果系列的电脑
软件环境：Xcode + Swift / Objective-C
账号：注册开发者账号，
并加入苹果开发者计划

学习iOS开发技能

请在App Store搜索并下载以下互动课程：
《Swift语言入门实例教程》
《Xcode互动教程 for iOS开发》
《Objective-C语言应用开发》
《App开发中的神兵利器》
注：196节SwiftUI课程包含在Xcode互动教程内

iOS开发教程
扫码免费下载

App Store的商业模式

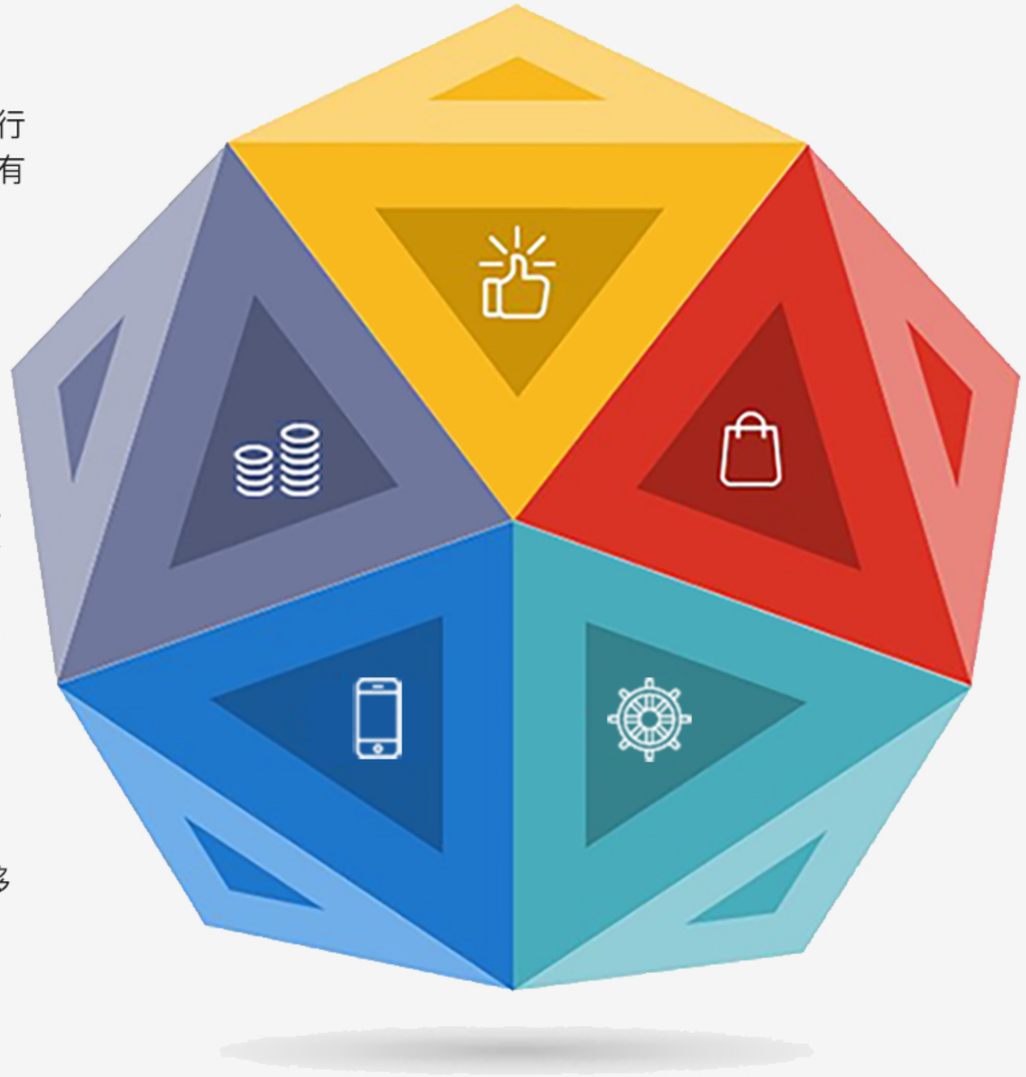
APP STORE'S BUSINESS MODEL

01 Who

主要的目标用户仍然是追求流行、时尚，对创作、学习、游戏有较强需求的客户群体。

02 What

作为连接开发者与用户的桥梁，一方面向用户提供安全、高质的服务，另一方面为开发者提供一个软件营销的权威平台。



03 Where

随着苹果手机和平板的火热销售，App Store现在已经覆盖了180个国家和地区，如中国、美国、英国、法国、德国、加拿大、日本等国家。

04 How

通过将App Store内嵌在每个移动设备，实现应用与终端的绑定，从而在全球拥有15亿左右的忠实用户群体。

05 Win

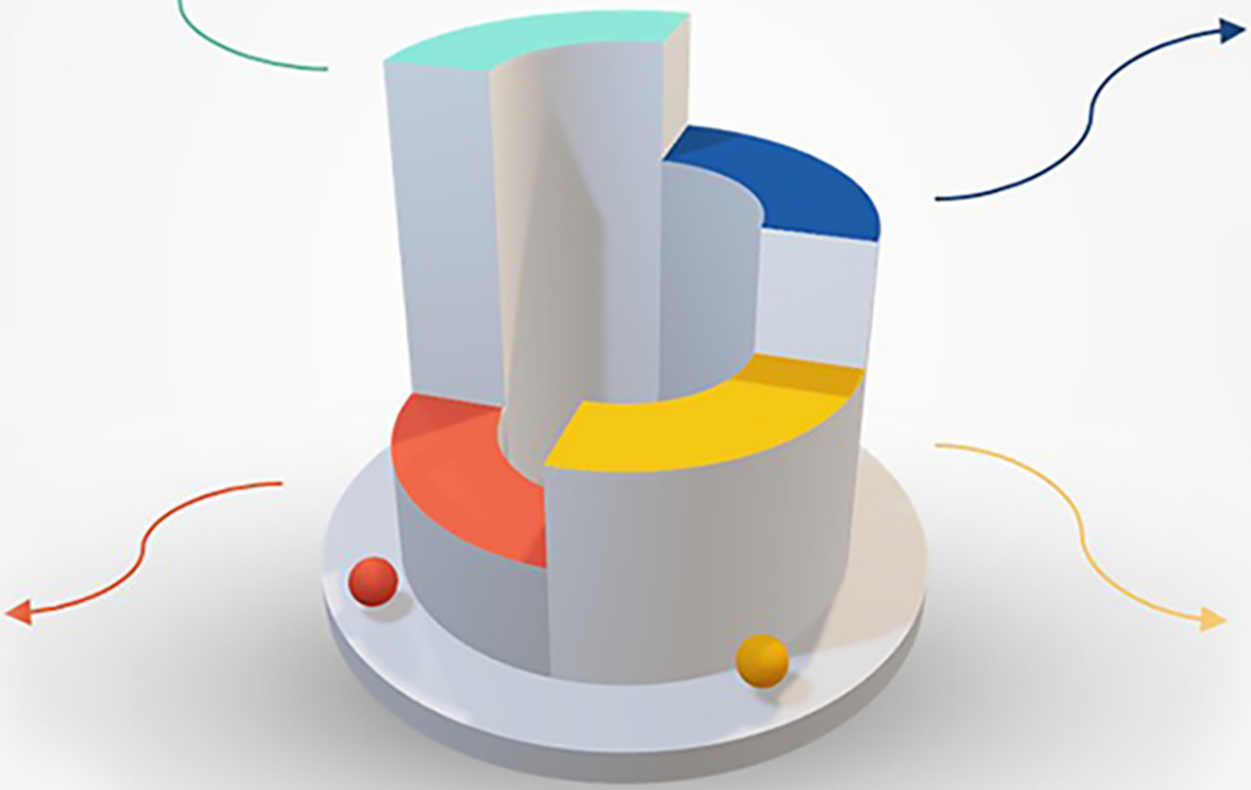
苹果公司将应用的收入按照三七的比例与开发者按月进行分成，即苹果公司获得收入的30%，软件开发者获得余下的70%。

App Store的盈利方案

THE APP STORE'S PROFITABLE PLAN

免费 + 内购

用户可以免费下载并使用App的部分功能，对于一些增值功能需要在App内，通过苹果的IAP渠道进行付费购买，这是很常见的一种盈利模式。



免费 + 广告

广告平台提供给开发者广告抽成，开发者只需在App中植入广告代码。用户可以免费下载并使用App，但是在场景开始或结束之后，需要观看弹出的广告。

完全免费

通常和线下服务搭配进行，为线下服务的提供便利。很多互联网巨头，也喜欢采用这种模式，主要用来迅速吸引客户和占领市场份额。

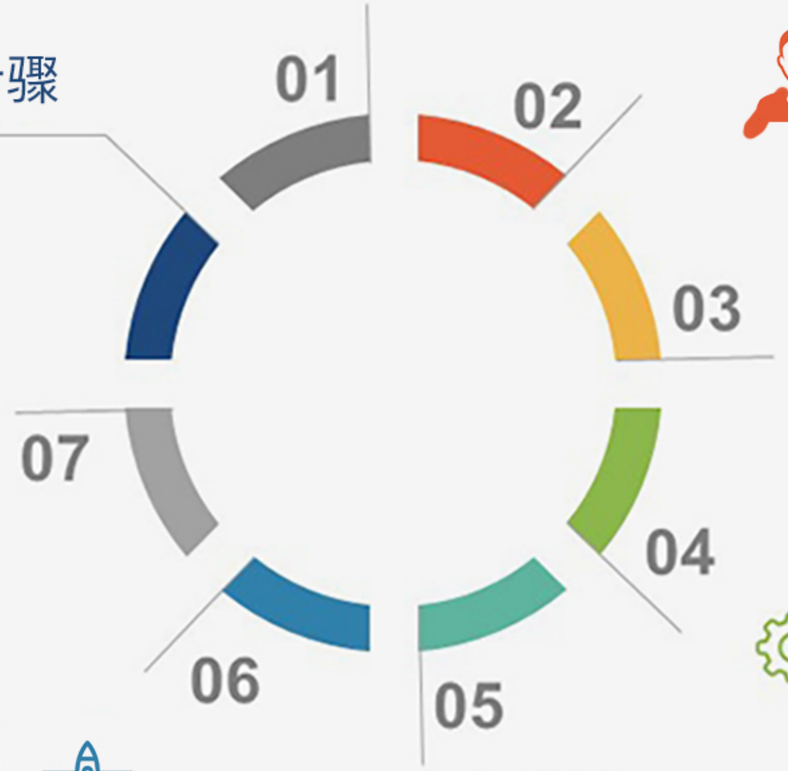
付费模式

用户需要付费才可以下载和使用App。这种模式比较适合精品应用或游戏，对自己的产品具有足够的信心！

如何入选App Store的精品推荐

HOW TO BE A BOUTIQUE IN THE APP STORE

入选精品推荐的七个步骤



UI设计

好的UI设计可以提升app 的易用性、吸引力和综合品质。

用户体验

开发者需要提高app的感官体验、交互体验、性能体验和情感体验。

创新

确保app 的功能、玩法具有新意，或者可以解决用户的特有难题。

辅助功能

除了具有完善的主要功能之外，还可以提供高度集成的辅助功能。

App Store 产品页面

是否拥有极具吸引力的屏幕快照、视频预览，以及贴切的描述文字。

独特性

和相同类型的app进行比较，自身的app要有独一无二的功能或玩法，例如我们推出的互动教程app。

本地化

通过高质量的翻译，让我们的app可以为全世界的用户提供相同品质的服务。

如何入选App Store的精品推荐 – 游戏篇

HOW TO BE A BOUTIQUE IN THE APP STORE

玩家参与度

参与度和留存率对于一个游戏来说具有非常巨大的意义，高的参与度表示这个游戏受到了玩家们的认可，是一款受欢迎的游戏。

游戏时的音频

精心设计的音乐和音效并不会让你意识到它的存在，它们会在潜意识里对你造成影响、缓慢而有效地带领你沉浸在游戏世界中。

游戏的操控性

以赛车游戏为例，操控性体现在赛车的启动和煞车是否灵敏、方向盘阻尼和转向控制的手感等。

游戏画面和性能

画面好和性能强从来不是对立的概念，又好看又好玩的游戏肯定最受广大玩家的欢迎。

叙事和故事性

对故事的渴求是人类的天性，人类从说故事和听故事中看到世界、看到自己。

可重玩性

不少游戏在制作上都比较注重这一点，游戏的难度合理、关卡精彩，其重复可玩性自然就高了。

App Icon的制作技巧

YOUR ICON IS THE FIRST OPPORTUNITY TO COMMUNICATE, AT A GLANCE, YOUR APP'S PURPOSE.

追求简洁

如果图标内容或形状过于复杂，则细节可能难以识别，尤其是在较小尺寸时。

一个中心

使用一个中心点设计图标，该图标可立即吸引注意力并清楚地识别应用。

可识别

设计一个美观而引人入胜的抽象图标，以可识别的艺术方式表达应用的目的。

背景简单

确保图标不透明，给图标一个简单的背景，并且不要使背景混乱。

不要包含照片、截图

照片细节很难在小尺寸下看到。屏幕截图对于图标来说过于复杂。

注重版权

不要使用 Apple 硬件产品的图标元素。Apple 产品受版权保护，不能在您的图标中出现。

测试场景

针对不同的壁纸测试您的图标在不同背景下的效果，不要只针对浅色或深色的壁纸进行测试。

方形图标

设计制作方形的图标，系统会自动给方形的图标添加圆角效果。

Launch Screen的制作建议

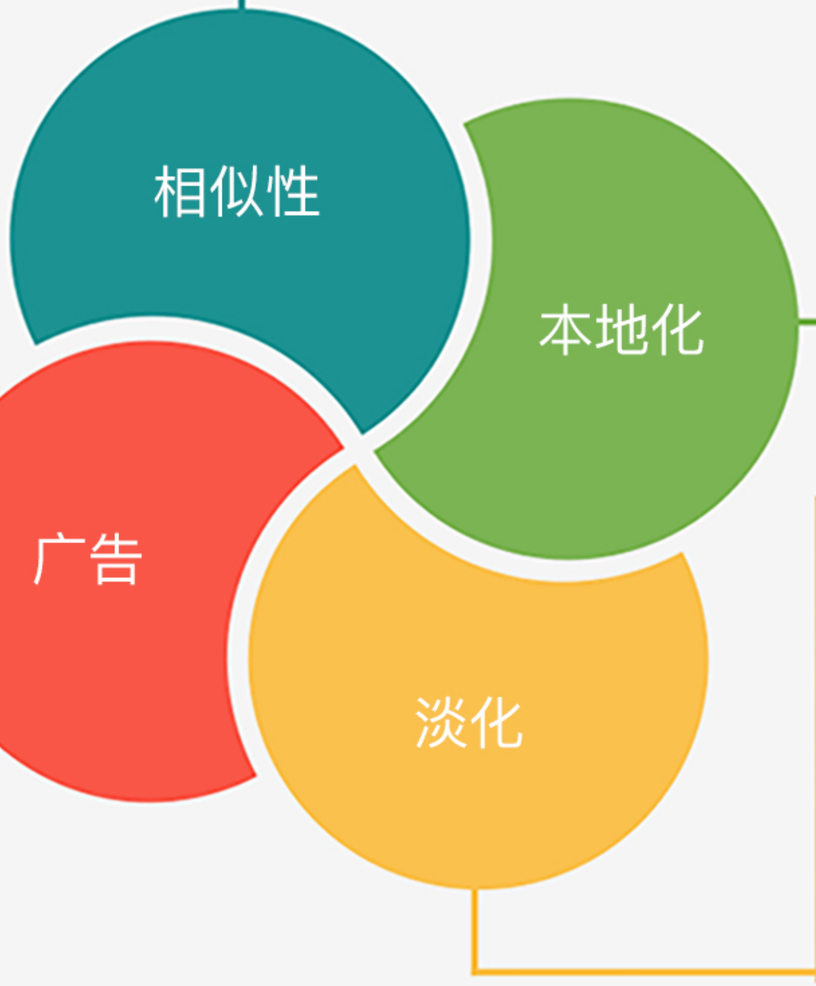
HOW TO IMPLEMENTING AN STORYBOARD LAUNCH SCREEN

设计和app首页相似的启动画面

如果启动画面和app首页画面差异较大，则在启动画面和app的首个画面之间，可能会产生令人不快的闪烁。

不要做广告

启动画面不是品牌的商机，不推荐在启动画面引入广告内容。



避免使用文本内容

由于启动屏幕是静态的，因此不会本地化显示文本，也就是说无法在启动画面对文本内容进行本地化。

淡化启动画面

用户可能会频繁切换应用，因此推荐设计一个不会引起对应用启动过于在意的启动画面。

提交App审核之前的准备工作

为了帮助您尽可能顺利地通过 app 审批，请查看下方列出的常见错误行为，
这些行为可能会导致审核流程延误或被拒。

01 测试

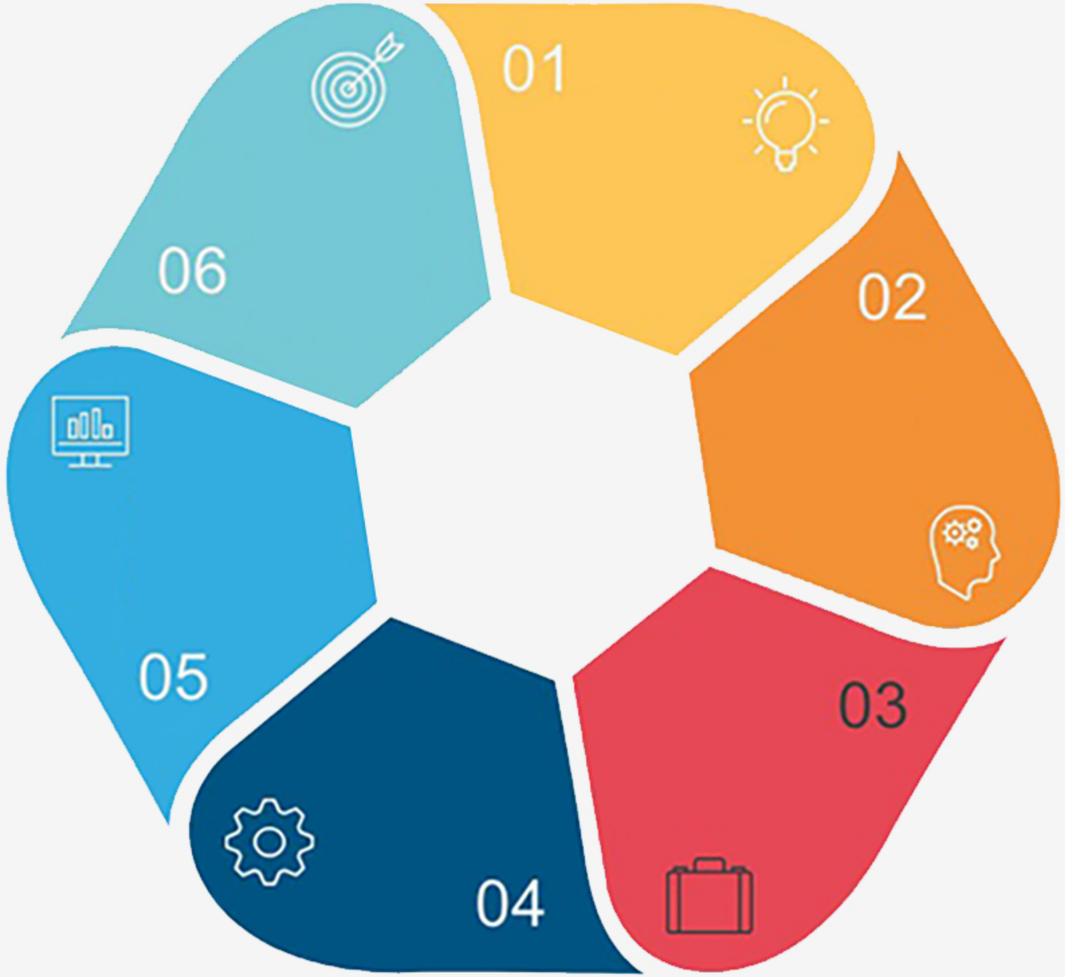
测试 app 是否会发生崩溃、是否存在错误。

02 完整性

确保所有 app 信息及元数据(meta data)完整且正确。

03 联系信息

及时更新您的联系信息，以便 App Review 部门在需要时与您取得联系。



04 演示帐户

如果您的app包含登录功能，需要提供演示帐户和密码以方便对app的审核。

05 后台服务

如果app拥有后台服务，确保后台服务已经启动，使其在审核期间处于可用状态。

06 备注

在 App Review 备注中附上与非明显功能，及 App 内购项目相关的详细说明。

App Store 审核指南 – 安全篇

当用户通过 App Store 安装 app 时，不应该出现如下的安全隐患：

令人反感的内容： App 不应包含具有攻击性、不顾及他人感受、令人不安、惹人厌恶、低俗不堪或只是让人感到毛骨悚然的内容。

数据安全： App 应确保遵守苹果开发者协议，妥善处理用户信息，防止对这些信息进行未经授权使用、或者被第三方访问。

用户生成的内容： 对于包含用户生成内容的 App，必须满足以下条件：
过滤令人反感的内容，以免在 app 中发布
制定一个机制，以举报攻击性内容
若用户发布攻击性内容，可以取消其使用服务的资格

开发者信息： 用户需要知道如何就疑问和支持与您取得联系。如果未能提供联系信息，不但会让客户有不好的感受，可能还会违反某些国家/地区的法律。

儿童类别的应用： 不得提供 app 外链接、购买机会或其他会对儿童造成干扰的内容，除非儿童位于受家长监控的指定区域中。

人身伤害： 例如医疗 app 可能会提供错误的数据或信息，此类 app 会被拒绝；App Store 也不允许分发任何鼓励消费烟草及电子烟产品的app。

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

App Store 审核指南 – 性能篇

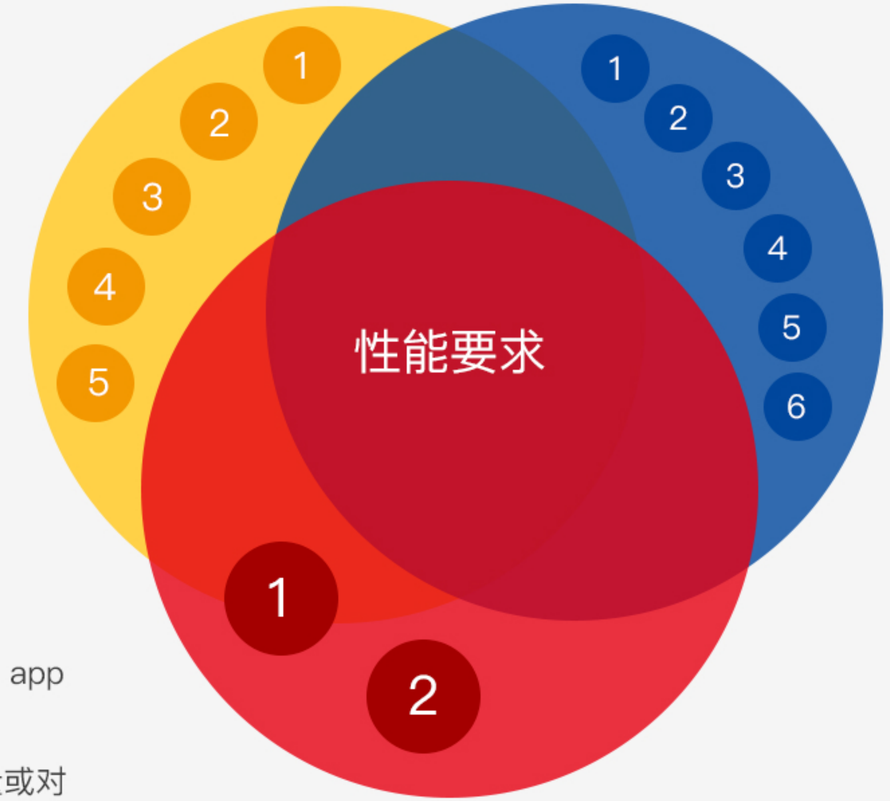
提交至 App Review 的申请应为 app 的最终版本，
并应包含所有必要的元数据和有效网址。

软件要求

1、App 仅可使用公共 API，并且必须能够在最新发布的 iOS 上运行。
2、应用不得在沙箱外读取或写入数据。
3、如果应用使用定位后台模式，请提醒用户，这么做会大幅降低电池续航能力。
4、如果应用需要浏览网页，则必须使用相应的 WebKit 框架。
5、对摄像头、麦克风等设备的使用，必须征得用户的同意，提供清晰的视觉或听觉指示。

硬件兼容性

1、为了确保用户能够充分利用您的 app，iPhone app 应尽量能在 iPad 上运行。
2、App 不应快速耗尽电池电能、产生过多的热量或对设备资源造成不必要的负担。



准确的元数据

1、请勿在 app 中隐藏某些功能，如果出现类似行为，则可能会从苹果开发者中除名。
2、如果您的应用包含内购项目，请确保应用的描述、预览清楚地指明是否有需要另行购买的精选项目、关卡和订阅等。
3、屏幕快照应展示 app 的使用情况，而非仅显示标题封面、登录页面或初始屏幕。
4、为了确保客户理解他们将在 app 中获得的体验，预览只可使用从 app 中采集的视频屏幕。
5、为 app 选择最适合的类别，如果类别与实际情况相差较远，苹果可能会更改 app 的类别。
6、选择一个独一无二的 app 名称，不要试图用商标术语、流行 app 的名称或其他不相关的短语来包装任何元数据。Apple 可以随时修改不合适的关键字，以防止不当的使用。

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

App Store 审核指南 – 商务篇

在 App Store 中，您可以通过多种方式让自己的 App 实现盈利

App 内购买项目

1. 您想要在 app 内解锁特性或功能等任何虚拟的物品，都必须使用IAP。
2. App 可以提供 App 内购买货币，供客户在 app 内“打赏”数字内容提供商。
3. 通过 App 内购买项目购买的所有点数和游戏货币等虚拟物品不得过期。
4. 您必须为所有可恢复的 App 内购买项目，设计一套恢复机制。
5. 提供随机虚拟物品购买的 app ，必须向客户披露每种类型物品的获取几率。

订阅

1. 订阅期必须持续至少七天，并且能够在用户的所有设备上访问。
2. 您可以在您的多个 app 中提供跨 app 的订阅，但不可扩展到第三方的 app。
3. App 不得强制要求用户为 app 评级或点评、才能访问该 app 的功能、内容。
4. 如果要将 app 更改为基于订阅的模式，您不得减掉现有用户已付费购买的功能。
5. 支持自动续期订阅的 app 可以通过提供 规定的信息，为客户提供免费试用期。

可以接受的其它业务模式

这里有一些额外的应做事宜您谨记在心

3、在租借期限结束后，禁止访问经批准的特定租借内容 (例如电影、音乐、图书)；所有其他项目服务不得存在过期时间。

4、钱包凭证可用于付款或接收付款、传输交易或是提供身份验证 (例如电影票、优惠券和 VIP 凭据)。

2、您可以在app中显示或推荐专为经批准的特定需求而设计的第三方 app (如健康管理、航空以及辅助功能等)。

5、保险类 app 必须免费提供，并且必须遵守 app 发布地区的相关法律，且不得使用 App 内购买项目。

1、在您的 app 中，可以出于购买或促销目的而展示您的其他 app，只要您的 app 不只是简单地罗列其他 app。

6、经批准的非营利组织可以在他们的 App 或第三方 app 内进行筹款活动，前提是这些筹款必须遵守所有的 App Review 准则并提供 Apple Pay 支持。

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

不可接受的业务模式

这里有一些不应做的事宜请谨记在心

2、人为地刷广告展示次数或者广告点击次数的 app，以及主要设计目的在于显示广告的 app。

4、强行限制 app 的用户群，例如限制特定地区或运营商。

2、通过由硬件或操作系统提供的内置功能（如推送通知、照相机或陀螺仪）或 iCloud 获利。

5、App 应当允许用户直接获得付费购买的项目而无需执行额外的任务，如在社交媒体上发帖、上传通讯录，以及签到特定次数等。

1、创建与 App Store 类似且用于显示第三方 app、扩展功能或插件的界面，或将其作为热门 app 的合集。

6、App 不得要求用户必须先为 app 评分或点评、观看视频、下载其他 app、点击广告等操作，然后才能访问 app 的功能、内容。

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

App Store 审核指南 – 设计篇

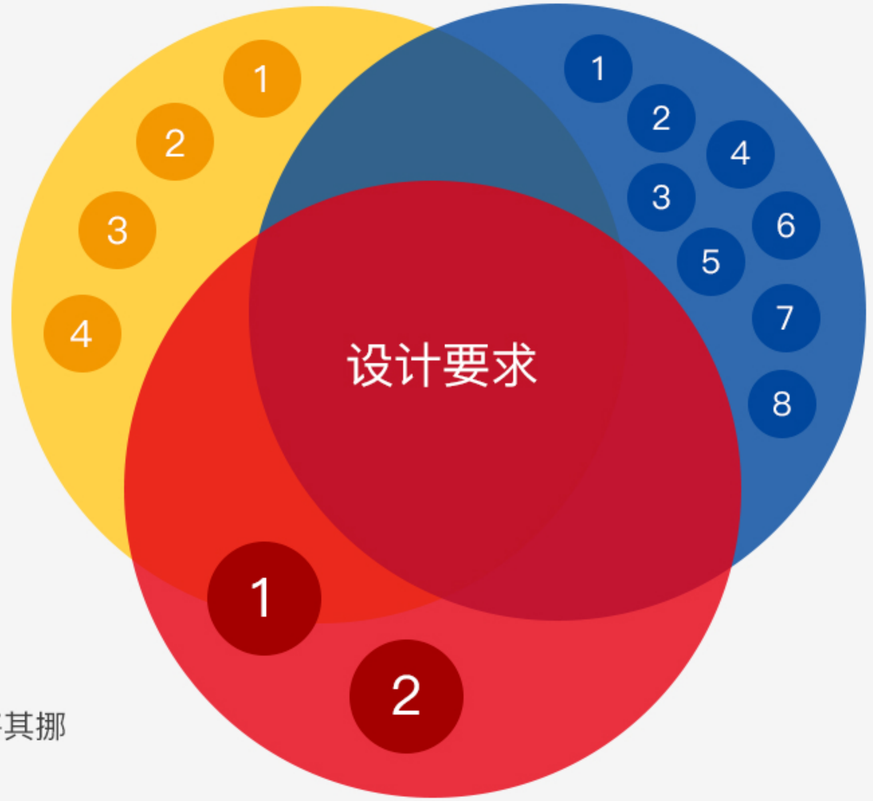
要想获准在 App Store 上发布 app，至少避免以下设计方面的错误：

重复 App

1、请不要为同一个 app 创建多个套装 ID。
2、如果您的 app 针对特定位置、机构等存在不同版本，请考虑提交单个 app，并提供 App 内购买项目以提供不同的功能。
3、避免在已有大量类似 app 的类别下开发；
4、上传大量相似版本 app 的开发者会遭到 Apple Developer Program 的除名。

抄袭者

1、不要简单照搬 App Store 上的热门 app。
2、不要只是细微修改其他 app 的名称或 UI，就将其挪为己用。



最低功能要求

1、App 应包含功能、内容和 UI，而不仅仅是一个经过重新包装的网站。
2、如果 app 没有什么实用价值、毫无新意、不能带来持久的价值，则可能无法获得批准。
3、使用 ARKit 的 app 应提供丰富而完整的增强现实体验，仅将模型放入 AR 视图或重播动画并不足够。
4、App 应能独立工作，无需安装其他 app。
5、如果 app 需要下载其他资源，请披露下载大小并在下载之前提醒用户。
6、如果 app 只是一首歌曲或一部影片，则应提交到 iTunes Store。
7、如果 app 只是一本图书或游戏指南，则应提交到 Apple Books Store。
8、利用商业化模板或 app 生成服务创建的 app 将被拒绝，除非app 由内容的提供商直接提交。

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

App Store 审核指南 – 法律篇

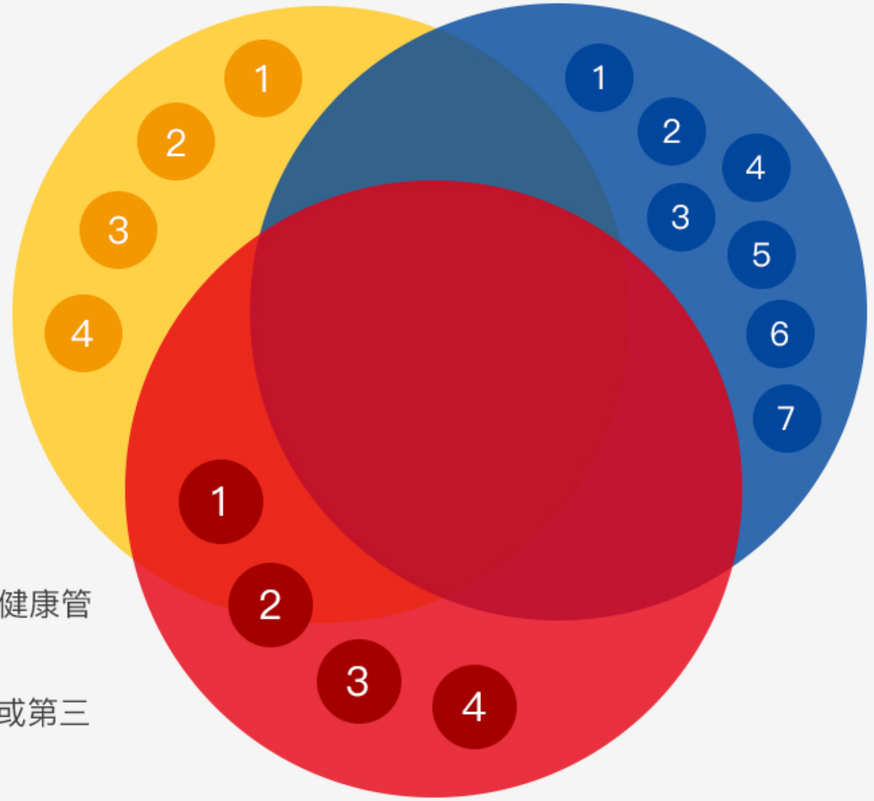
只要 app 向某个地区的用户提供，那么就必须遵守该地区的所有法律要求

数据使用和共享

1、除非法律另有许可，否则您不得未经他人允许而使用、传输或共享他们的个人数据。
2、除非法律另有许可，否则未经用户额外同意，为一个用途而收集的数据不可用于其他用途。
3、App 不得试图基于收集的数据构建用户资料
4、从 HomeKit API、HealthKit、ARKit、相机 API 等收集的数据，不得用于市场营销、投放广告或基于使用情况进行其他数据挖掘。

其它法务信息

1、App 不得将虚假或错误数据写入任何医疗研究/健康管理 App，不得在 iCloud 中存储个人健康信息。
2、主要面向儿童的 app 不应包含第三方数据分析或第三方广告。
3、请确保 app 只包含由您创建或拥有使用许可的内容。
4、不得在未经授权的情况下，在 app 中使用受保护的第三方材料 (例如商标、版权作品、专利设计)；



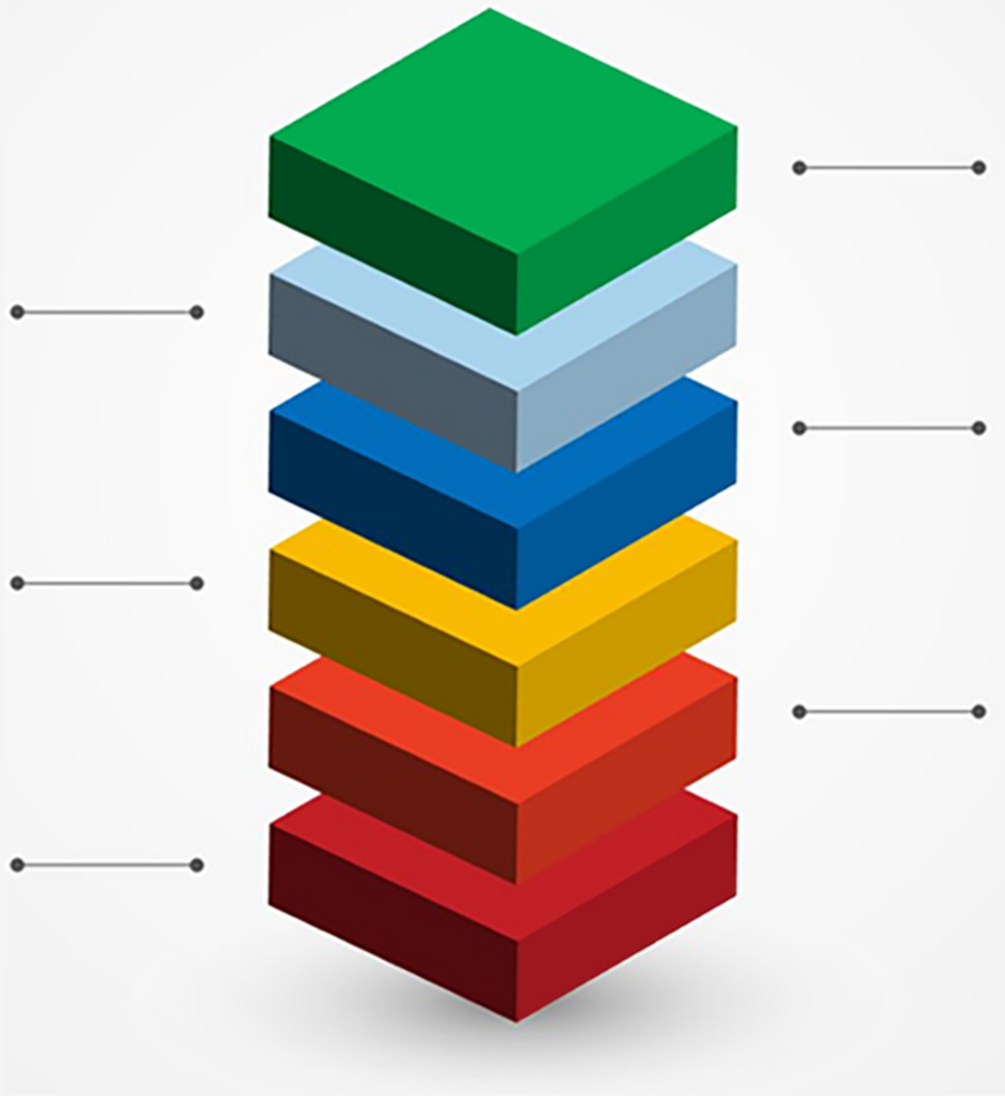
数据收集和存储

1、在 Apple 生态体系中，保护用户隐私总是第一要务。
2、所有 app 必须在 App Store Connect 元数据栏位包含可轻松访问的隐私政策链接。
3、如果 app 会收集用户数据或使用数据，app 必须征得用户的同意才能收集。
4、App 仅可请求访问与 app 核心功能相关的数据，并且仅可使用完成相关任务所需的数据。
5、App 必须尊重用户的权限设置，不得操纵、欺骗或强迫用户同意不必要的数据访问。
6、如果 app 不包含基于帐户的重要功能，请允许用户在不登录的情况下使用。
7、如果开发者的 app 试图暗中收集用户密码或其他用户私人数据，那么开发者会被从 Apple Developer Program 中除名。

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

开发者行为准则

客户的信任是 App Store 获得成功的基石，所以 App 需要避免以下行为：



6、在 app 内部或外部实施任何其他操纵行为

5、针对未交付的功能或内容收取费用

4、以欺骗的方式抬高价格

3、强迫用户共享不必要的数据

2、诱导用户进行非自愿的购买

1、掠夺用户或试图勒索用户

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

如何有效地在App的Sandbox中存储数据

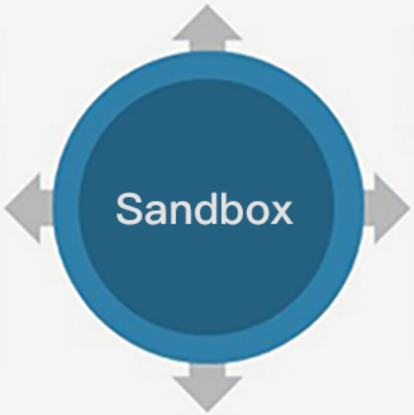
iOS DATA STORAGE GUIDELINES

<Application_Home>/Documents目录

只有用户生成的内容，应存储在Documents目录中，并由iCloud 自动备份。例如一个记事本应用，用户所写的日志，可以保存在 Documents 目录里。

<Application_Home>/Library/Caches目录

可以再次下载或重新生成的数据应存储在Caches目录。例如一个新闻应用，如果需要从服务器下载内容展示给用户看，下载的内容就可以放在该目录里。



<Application_Home>/Library/Preferences目录

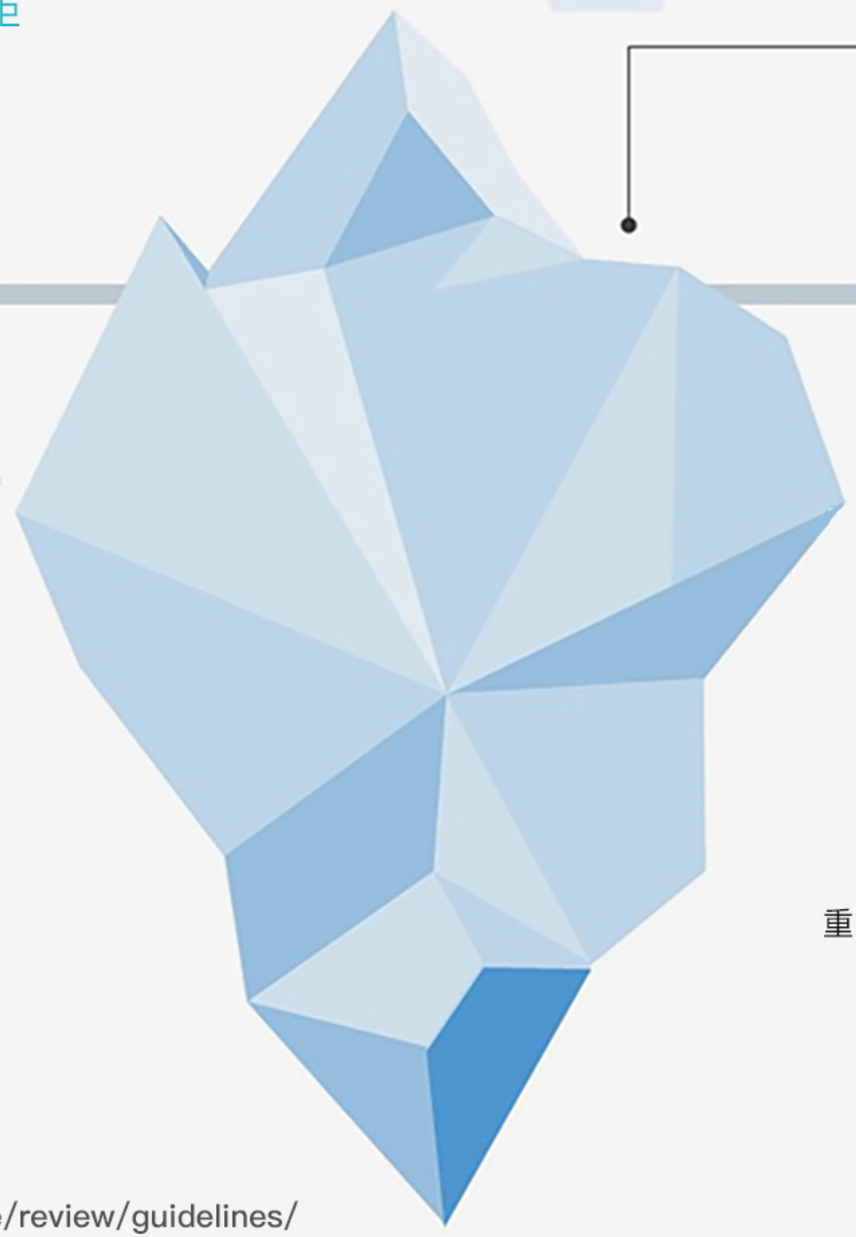
该目录包含应用程序的偏好设置文件。您不应该直接创建偏好设置文件，而是应该使用UserDefaults类来取得和设置应用程序的偏好.

<Application_Home>/tmp目录

仅临时使用的数据应存储在tmp目录中，尽管这些文件未备份到 iCloud，但请记住，在使用这些文件后，请将其删除，以免它们继续占用用户设备上的空间。

避免常见的 App 拒绝情况

超过40%是因“指南 2.1–性能: App完成度”遭拒



成功上线！

务必在运行最新版系统的设备上对应用进行全面测试并修复所有错误后再提交。

崩溃和错误

所有应用都需要提供完善可用的用户支持链接，以及隐私政策链接。

链接损坏

仍处于开发中和包含占位符内容的 app 不能进行发布，且无法获得批准。

占位符内容

确保 app UI 和产品图像与 App Store Connect 中对应的设备类型相符。

屏幕快照不准确

信息不完整

如果某些功能需要登录，请提供有效的演示帐户用户名和密码。

UI不合标准

Apple 非常注重简洁、雅致且用户友好的界面。

重复提交类似 App

请周全地将您的 app 整合成一个，以改善您的审核体验，以及未来用户的体验。

持久价值不足

如果 app 缺乏足够的功能或内容，或者只适用，则可能无法获得

参考资料：https://developer.apple.com/cn/app-store/review/guidelines/

iOS应用开发的八个特点

在开发运行在移动平台上的应用时，需要注意和传统的软件开发的一些不同之处。

1、屏幕的不同

开发者需要把显示给用户的内容，合理地组织在一块小小的屏幕上，所以需要设计者进行精心的设计和排版。

2、交互方式不同

iOS系统采用手指触摸的方式进行人机交互，所以要尽可能使按钮等交互控件的尺寸保持在44点以上，以避免误操作。

3、内存的不同

移动设备内存通常在1G～4GB之间，需要在应用中合理使用媒体资源，避免因太耗内存而被系统关掉。

4、电量的不同

及时关闭地理定位服务，减少不必要的网络请求，尽量避免以轮询的方式工作，不然会使CPU无法进入睡眠状态，从而引起电量的长时间消耗。

5、安全方面的限制

一个App作为一个程序束bundle存在，App只可以访问其资源束内的文件夹或资源文件。

6、可访问的设备众多

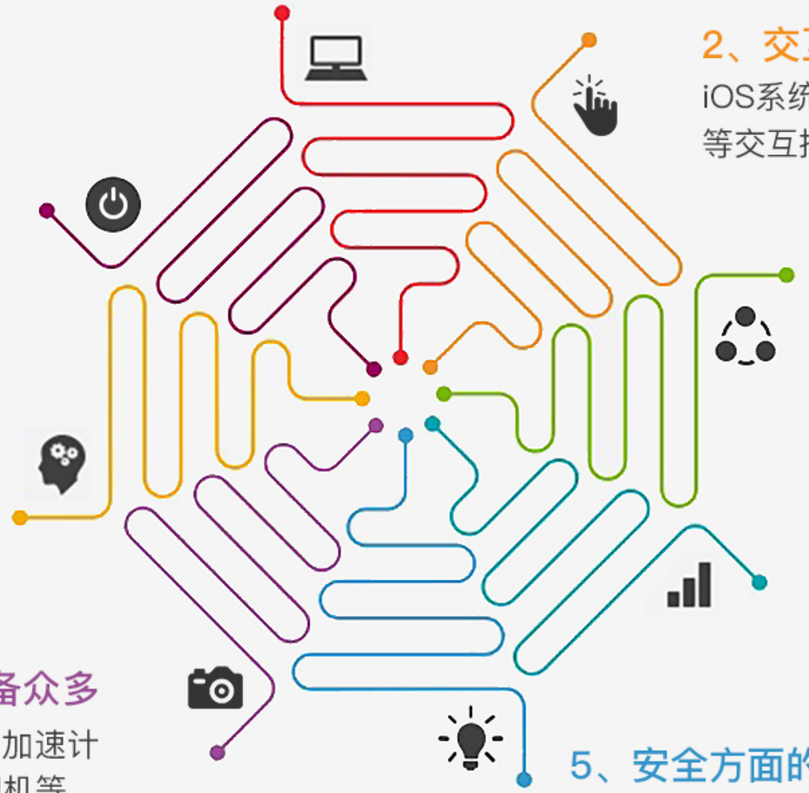
在iOS中运行的应用，可以访问设备自带的加速计、陀螺仪、定位设备、蓝牙、相机等。

7、下拉菜单

很少使用菜单进行页面跳转，通常采用导航控制器或标签控制器进行页面间的导航。

8、程序退出方式的不同

没有关闭按钮，用户通过底部的Home键或者手势，退出运行的应用，并且应用退出后，仍然会在内存中保留一段时间。



使用UIApplicationDelegate管理生命周期

RESPOND TO APP-BASED LIFE-CYCLE EVENTS

1、Not Running

此时app处于未运行的状态。

3、Background

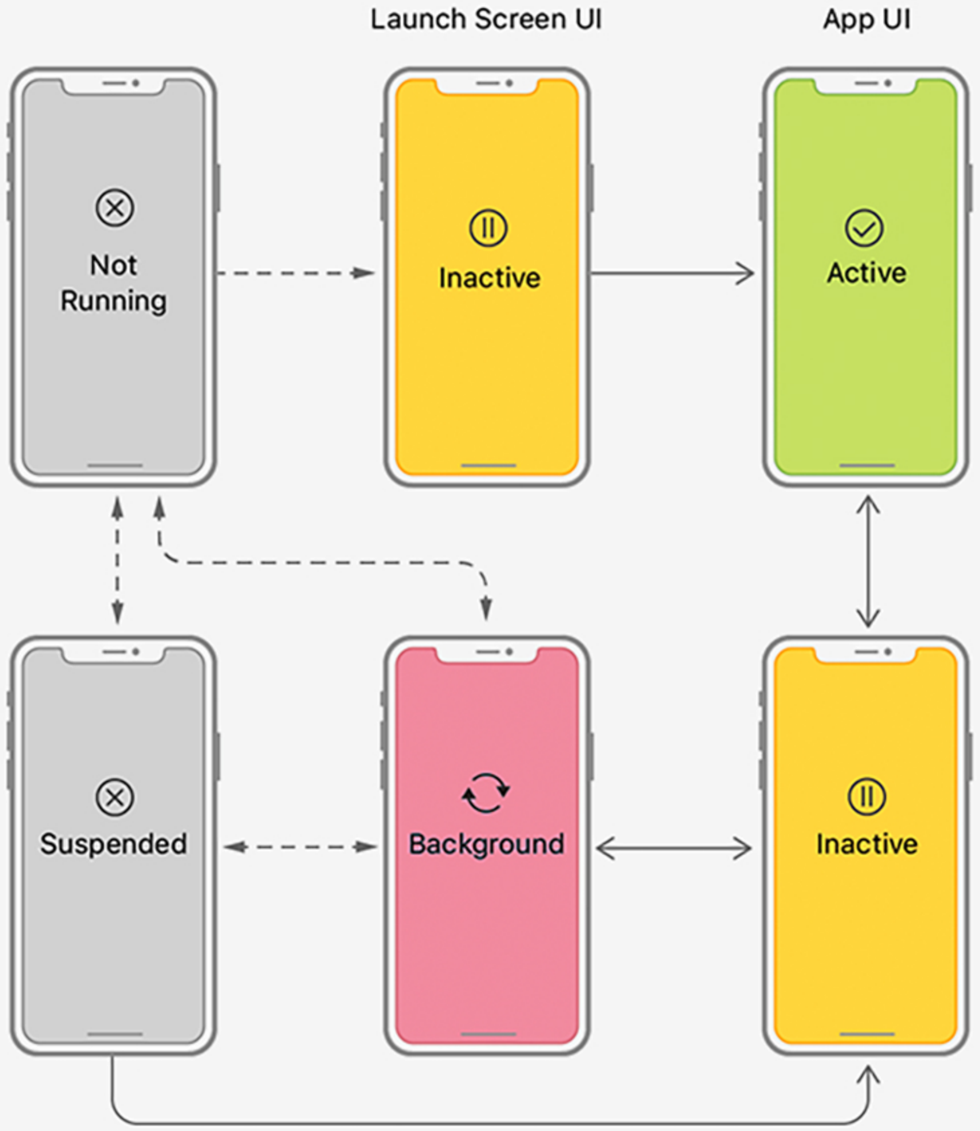
系统请求通常移动到后台处理，以便它可以处理事件。例如，系统可能会在后台启动场景以处理位置事件。

5、Suspended

处于背景状态的app，最终会被系统移动到挂起状态，此时app不再执行代码。

6、

之后，状态在活动和后台之间来回切换，直到应用终止。调用applicationWillTerminate:可以在退出前，保存用户的一些重要数据。



2、Foreground Active

当app启动时，首先由not running状态切换到inactive状态，此时调用application:didFinishLaunching-WithOptions:方法；然后由inactive状态切换到active状态，此时调用applicationDidBecomeActive:方法

4、Background

当用户关闭应用的 UI 时，app会切换到背景状态。此时调用applicationWillResignActive:方法；然后从非活动状态切换到背景状态，此时调用applicationDidEnter-Background:方法。

参考资料：https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle

使用UISceneDelegate在基于场景的app中管理生命周期

RESPOND TO SCENE-BASED LIFE-CYCLE EVENTS

1、Unattached

当用户或系统请求应用的新场景时，UIKit 将创建它并将其置于未连接状态。

3、Background

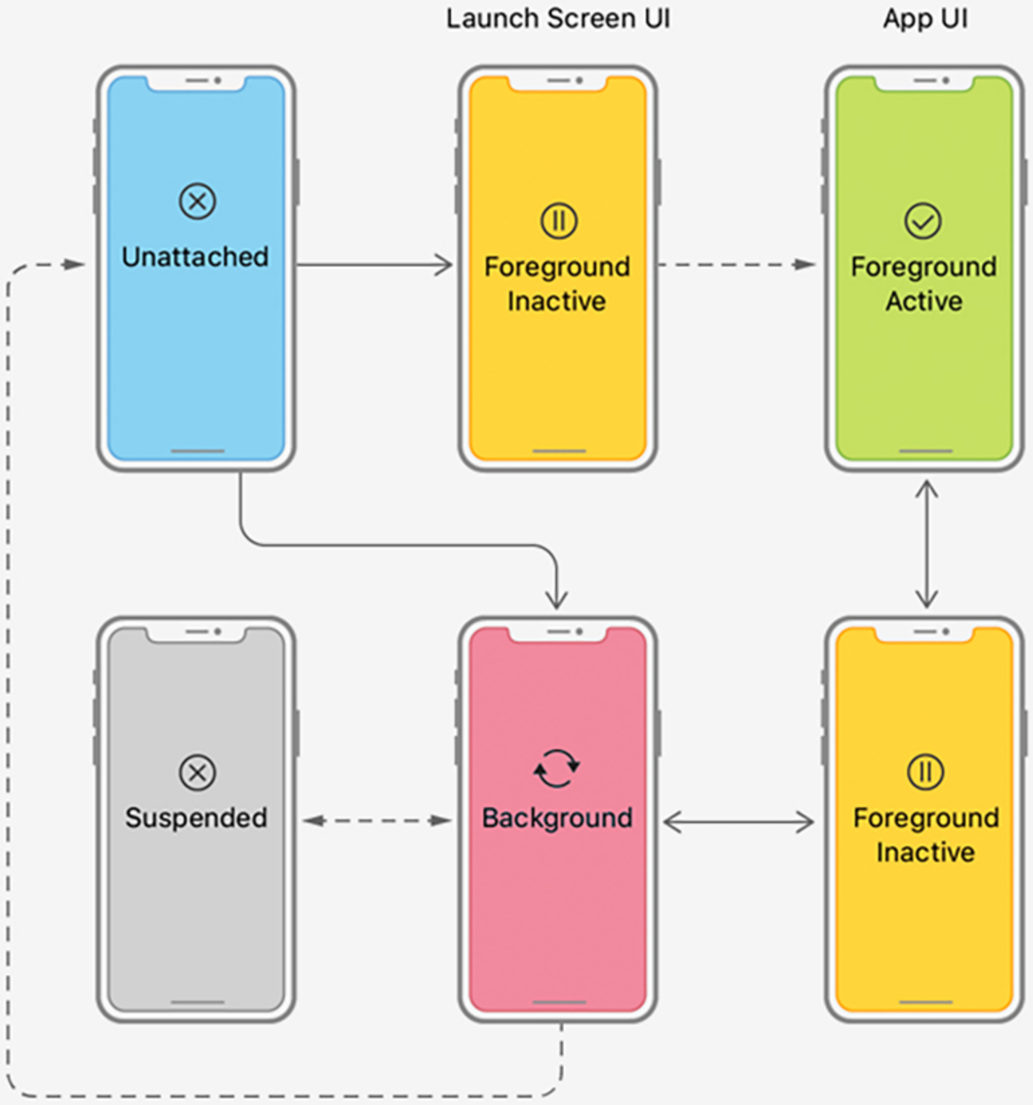
系统请求的场景通常移动到后台，以便它可以处理事件。例如，系统可能会在后台启动场景以处理位置事件。

5、Suspended

处于背景状态的场景，最终会被系统移动到挂起状态。

6、Unattached

UIKit 可以随时断开背景或挂起的场景以回收资源，将该场景返回到未附加状态。



2、Foreground Active

用户请求的场景会快速移动到前景，它们出现在屏幕上。

4、Background

当用户关闭应用的 UI 时，UIKit 会将关联的场景移动到背景状态。

参考资料：https://developer.apple.com/documentation/uikit/app_and_environment/managing_your_app_s_life_cycle

关于应用程序的启动序列

LEARN THE ORDER IN WHICH YOUR CUSTOM CODE IS EXECUTED AT LAUNCH TIME

1、应用由用户显式启动，或由系统隐式启动。

2、Xcode 提供的main函数调用 UIKit 的 UIApplication-Main(_:_:_:_:) 功能。

3、UIApplicationMain(_:_:_:_:) 函数创建UIApplication和应用委托。

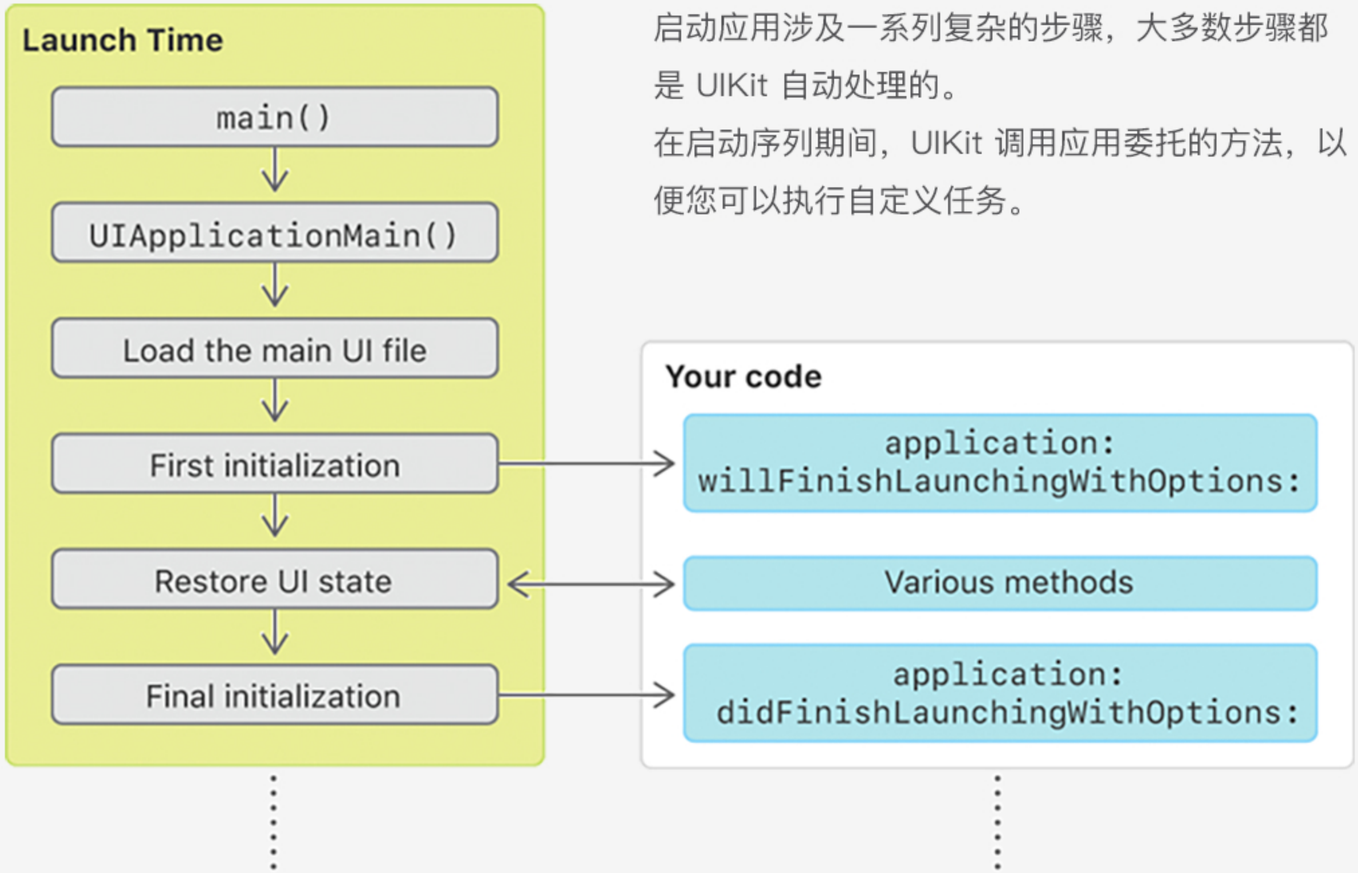
4、UIKit 从主故事板或xib文件加载应用的默认接口。

5、UIKit 调用delegate的application(_:willFinishLaunching-WithOptions:) 。

6、UIKit 执行状态还原，这将调用应用委托和视图控制器的其他方法。

7、最后 UIKit 调用delegate's application(_:didFinishLaunch-ingWithOptions:)

8、初始化完成后，系统使用场景委托(scene delegates)或应用委托(app delegate)显示 UI 并管理应用的生命周期。



启动应用涉及一系列复杂的步骤，大多数步骤都是 UIKit 自动处理的。

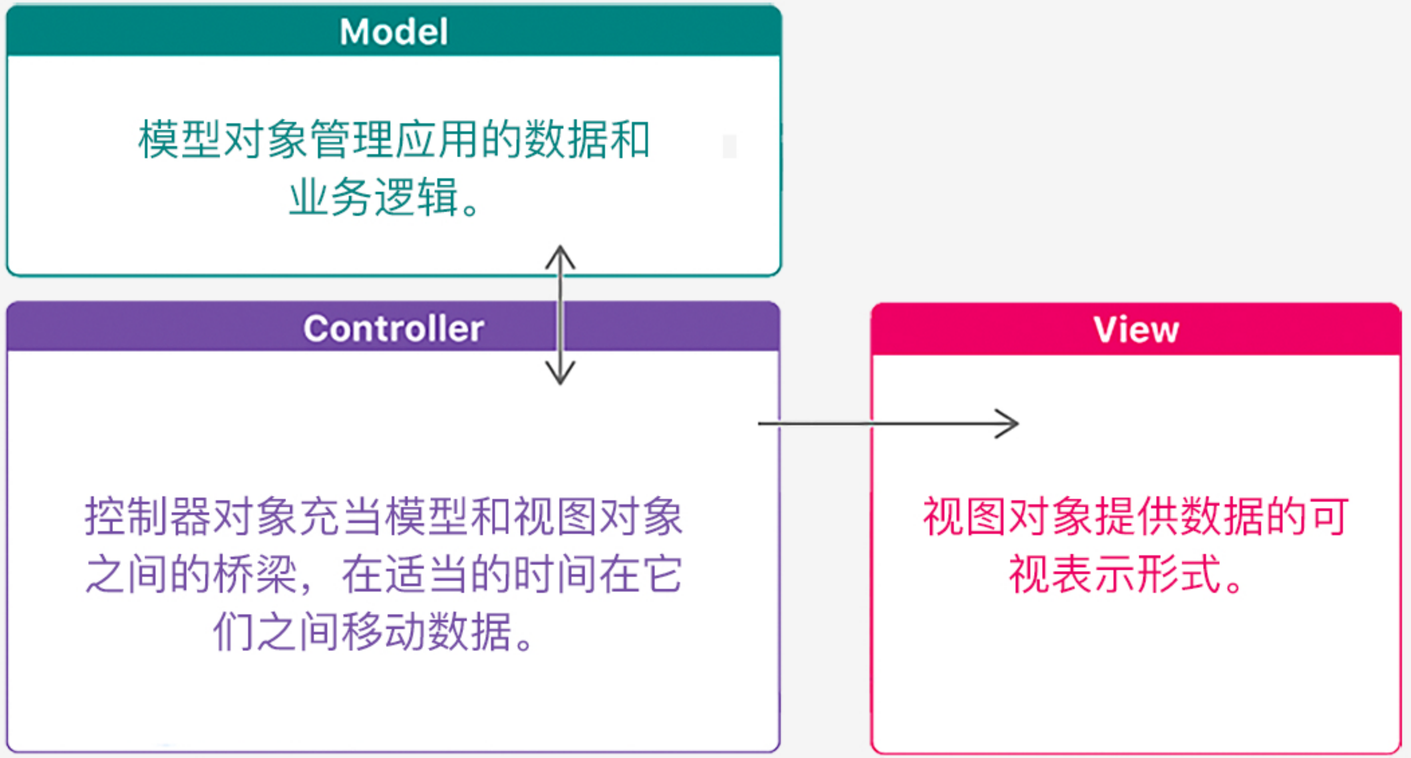
在启动序列期间，UIKit 调用应用委托的方法，以便您可以执行自定义任务。

参考资料：https://developer.apple.com/documentation/uikit/app_and_environment/responding_to_the_launch_of_your_app/about_the_app_launch_sequence

UIKit应用中的MVC代码结构

CODE STRUCTURE OF A UIKIT APP

UIKit 应用的结构基于模型–视图–控制器(MVC)设计模式，其中对象按其用途进行划分。



参考资料：https://developer.apple.com/documentation/uikit/about_app_development_with_uikit

UIKit应用中的MVC代码结构

CODE STRUCTURE OF A UIKIT APP

Mdoel模型：

1、你来提供表示应用数据结构的模型对象。

2、UIKit 提供了一个 UIDocument 对象，用于组织属于基于磁盘的文件的数据结构。

3、Foundation 框架定义表示字符串、数字、数组和其他数据类型的基本对象。

View视图：

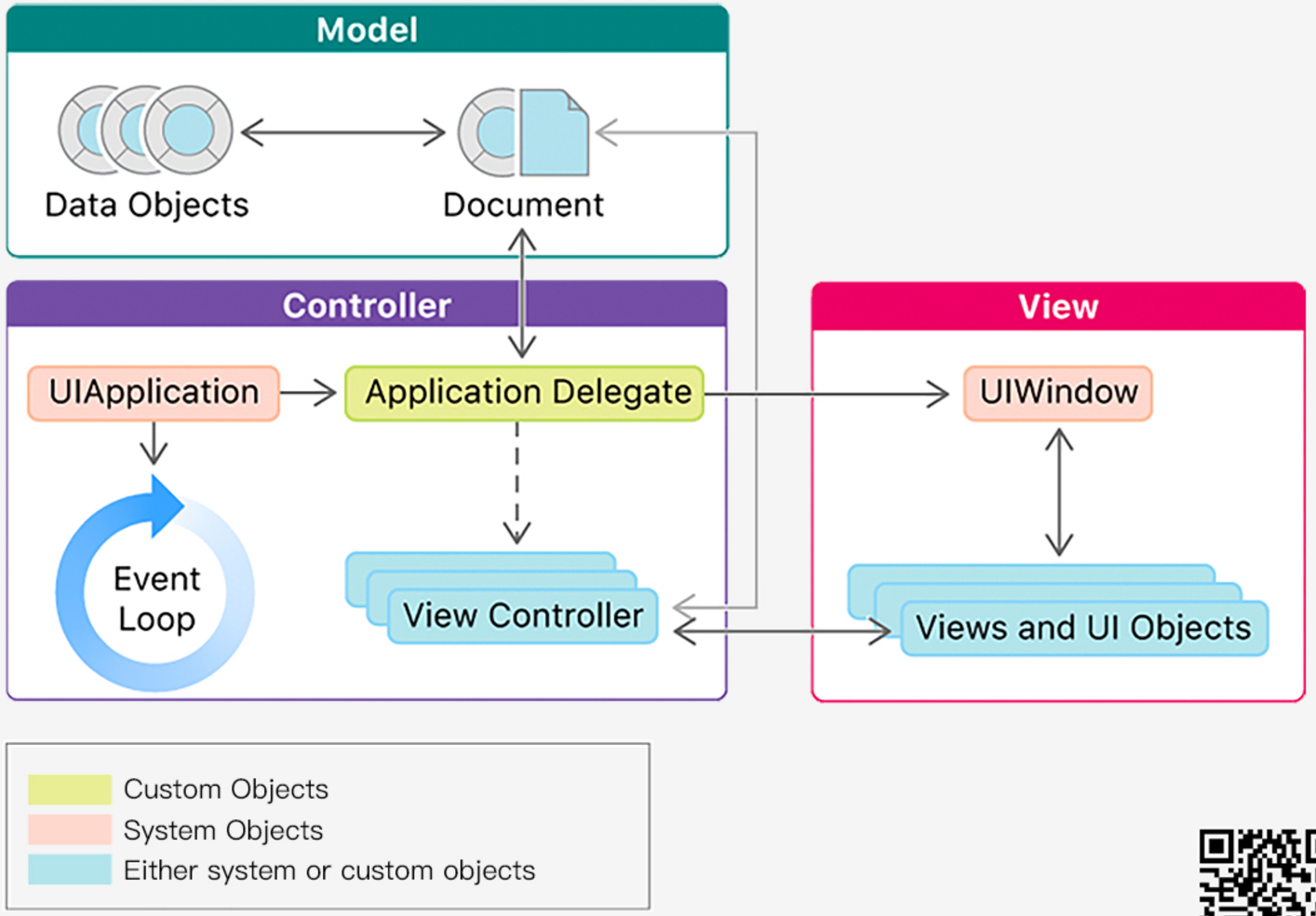
1、UIKit 提供了大多数视图对象，但您可以根据需要为数据定义自定义视图。

2、UIKit定义了UIView类，它通常负责在屏幕上显示您的内容。

Controller控制器：

1、协调数据对象和 UIKit 视图之间的数据交换是视图控制器和应用委托对象。

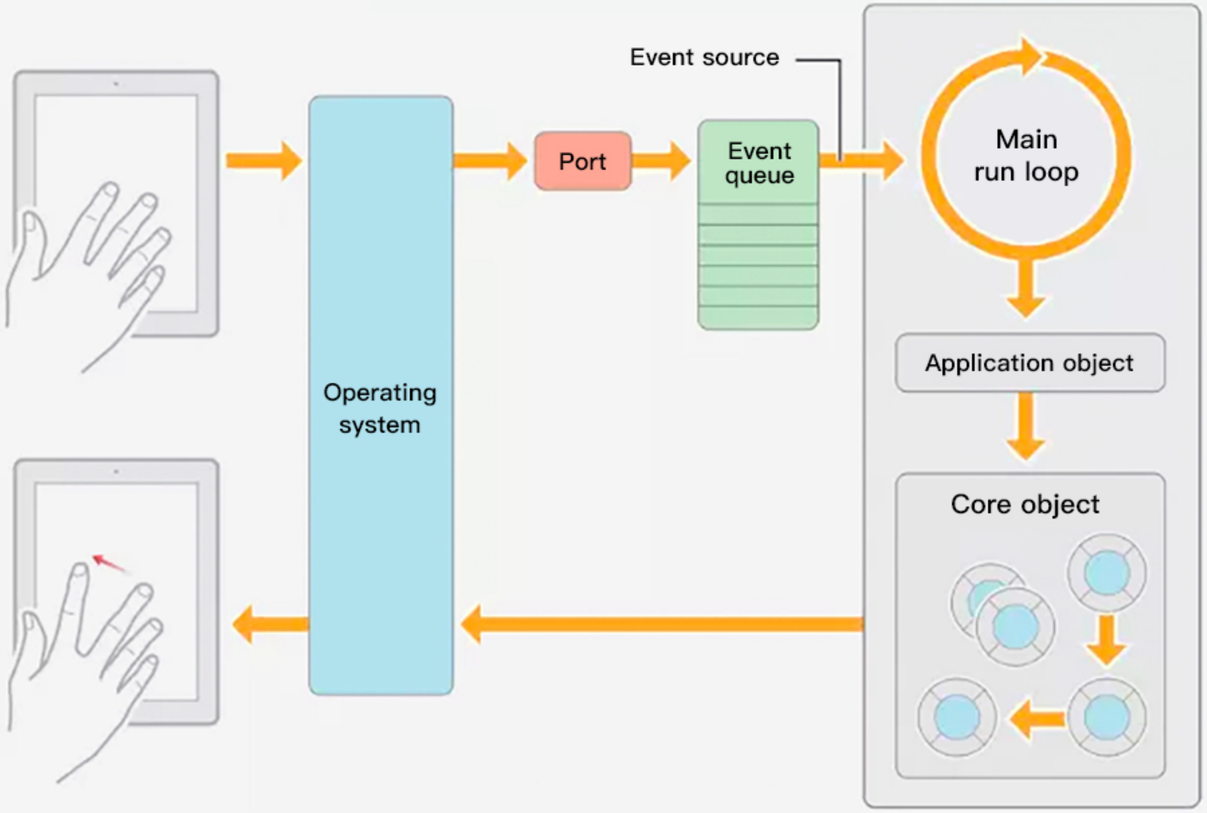
2、UIApplication对象运行应用的主事件循环，并管理应用的总体生命周期。



参考资料：https://developer.apple.com/documentation/uikit/about_app_development_with_uikit

使用Main run loop处理与用户相关的事件

PROCESSING EVENTS IN THE MAIN RUN LOOP



Processing events in the main run loop

main run loop主要作用是处理所有与用户相关的事件。

main run loop运行在应用程序的主线程，从而确保接收到的用户相关的事件，可以被有序地处理。

1、当用户与设备交互时，系统就会生成与交互相关的事件。

2、然后事件被UIKit通过一个特殊的端口来分发。

3、应用程序把事件放入队列，然后再逐个分发到main run loop来执行。

4、UIApplication对象从Main run loop接收传来的事件.

5、然后UIApplication将事件交给控制器、视图等核心对象来处理。

6、大多数的事件通过main run loop来分发，但有些不是，这些事件被发送到一个delegate对象或传递到block中。

参考资料：https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/Multithreading/RunLoopManagement/RunLoopManagement.html

关于屏幕、窗口和视图

UISCREEN, UIWINDOW AND UIVIEW

UIScreen

1、每个设备至少有一个 UIScreen 对象表示设备的主屏幕，其他屏幕对象表示连接的显示器。

2、窗口对象充当应用在屏幕上内容的容器，屏幕向应用负责基础显示的特征。

UIWindow

1、UIWindow 本身不提供可见内容。UIWindow 的所有可见内容都由其根视图控制器提供，您可以在Storyboard中配置该控制器。

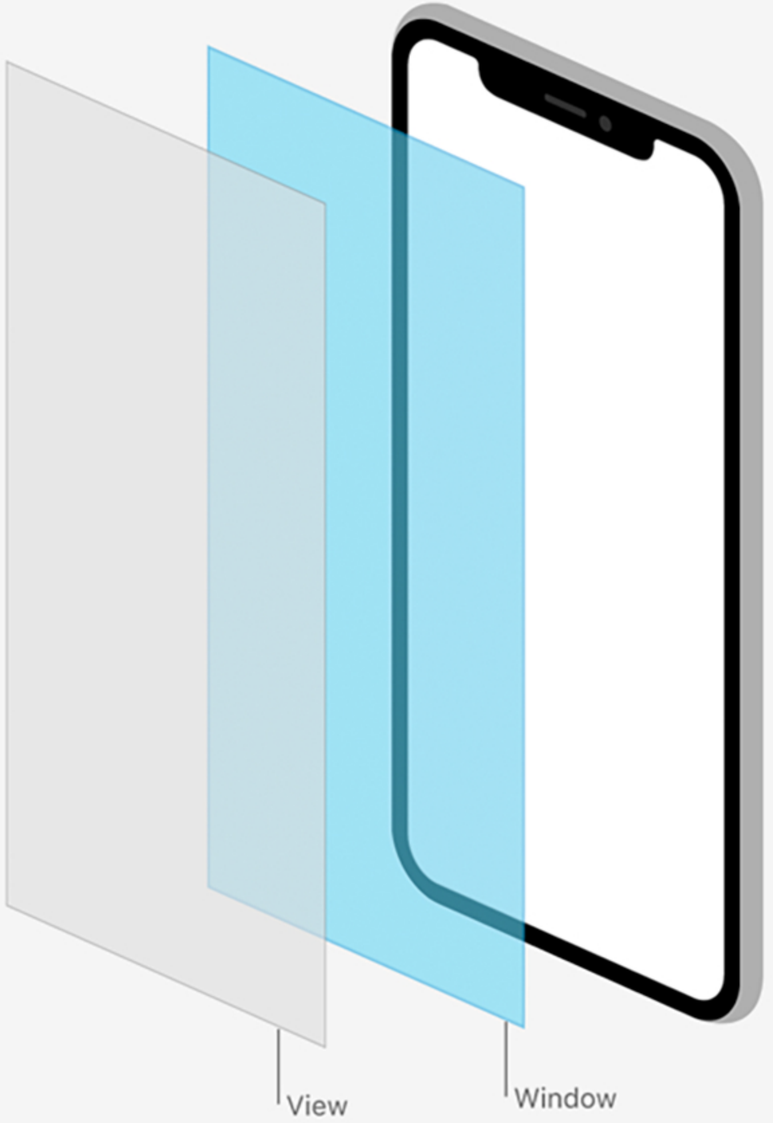
2、该窗口的作用是从 UIKit 接收事件，并将任何相关事件转发到根视图控制器和相关视图。

3、UIKit 提供了一个初始窗口，您可以根据需要创建其他窗口。

UIView

1、视图是应用UI的基本构建基块，UIView 类定义所有视图共有的行为。

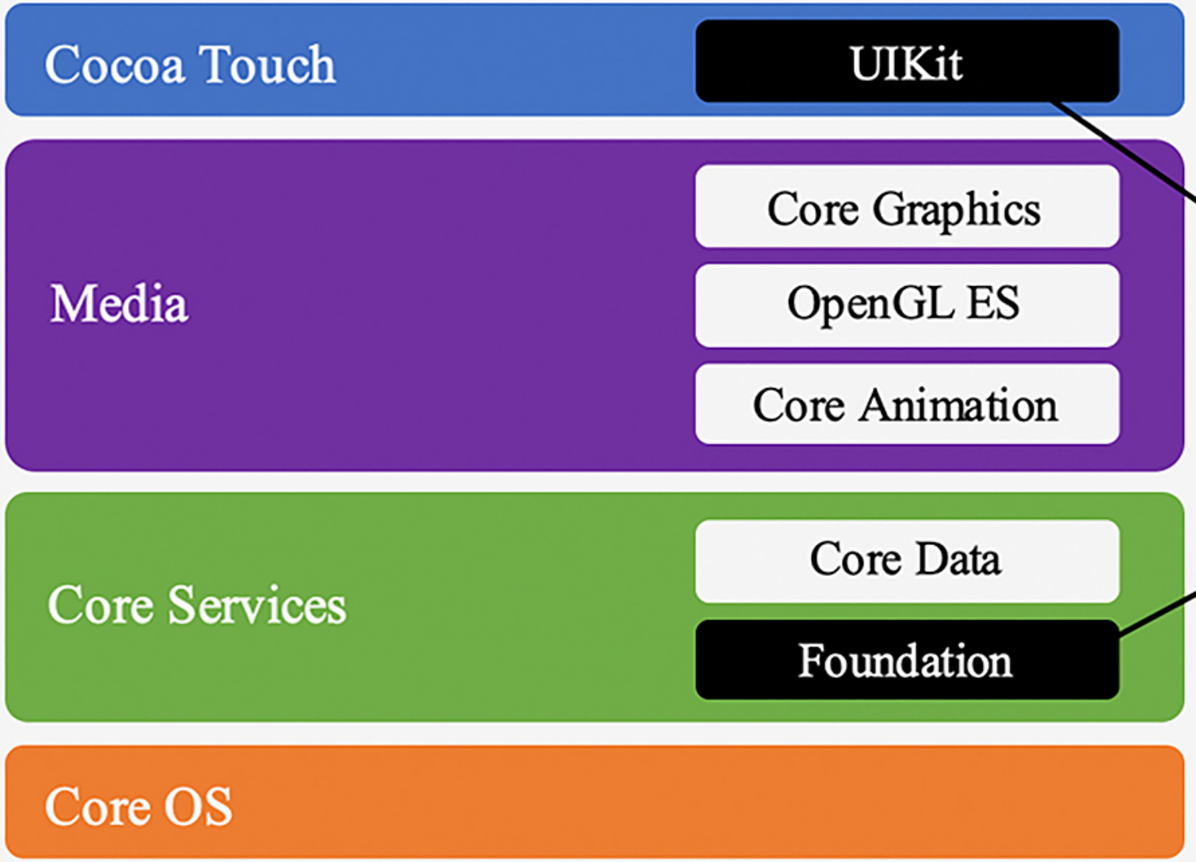
2、视图对象在其边界矩形内呈现内容，并处理与该内容的任何交互。



参考资料：https://developer.apple.com/documentation/uikit/windows_and_screens

UIKit框架和Foundation框架

Cocoa框架包含了众多子框架，其中最重要的是“Foundation”和“UIKit”两个框架。



Core Frameworks

UIKit框架是基础的UI类库，以后我们在iOS开发中会经常用到。

Foundation是框架的基础，和界面无关，主要包含了大量常用的API，如值对象、集合、文件系统、URL、进程通讯、通知、归档和序列化、表达式和条件判断。

应用程序中的事件响应链

RESPONDER CHAINS IN AN APP

右图包含标签、文本输入框、按钮和两个背景视图，当用户使用手指点击文本输入框时的事件响应链如下：

1、如果文本输入框不处理事件，UIKit 会将事件发送到文本输入框的父 UIView 对象；

2、如果父 UIView 不处理事件，UIKit 会将事件发送到控制器的根视图；

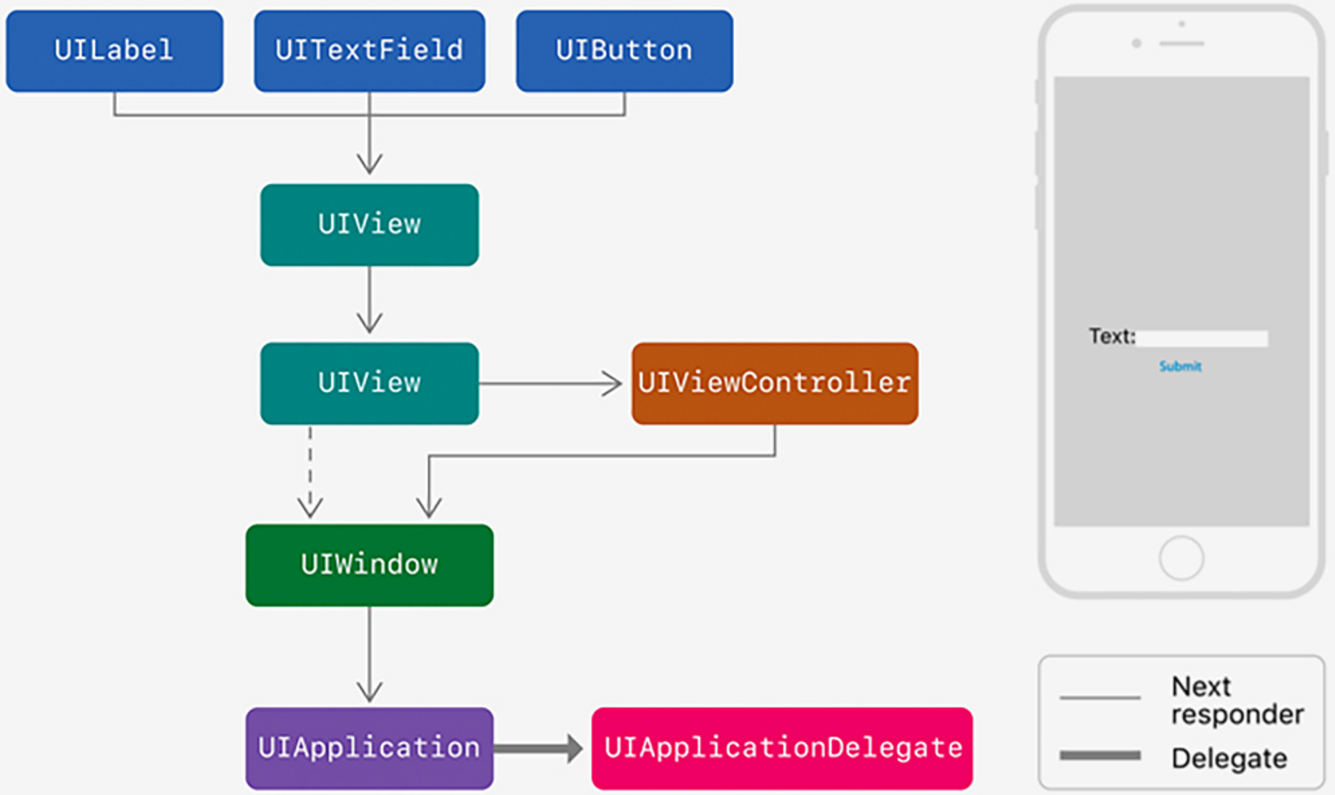
3、响应器链在将事件定向到窗口之前，将事件转移到根视图所属视图控制器。

4、如果视图控制器不响应事件，UIKit 会将事件传递到窗口对象。

5、如果窗口也无法处理该事件，UIKit 会将事件传递到 UIApplication 对象。

6、如果该对象是 UIResponder的实例，并且不是响应器链的一部分，则可能会将该事件传递到应用委托。

7、如果应用委托也无法处理该事件，则事件将被丢弃。



参考资料：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/using_responders_and_the_responder_chain_to_handle_events

触摸事件的四个阶段

THE PHASES OF A TOUCH EVENT

Touch begin

当手指或苹果笔触摸屏幕时，UIKit 将创建一个 UITouch 对象，将触摸位置设置到适当的点，并将其相位属性设置为 UITouch.Phase.began。

Touch moved

当同一手指在屏幕上移动时，UIKit 会更新触摸位置并将触摸对象的相位属性更改为 UITouch.Phase.move。

Touch ended

当用户从屏幕上抬起手指时，UIKit 会将相位属性更改为 UITouch.phase.end，触摸序列结束。

Touch canceled

同样，系统可能随时取消正在进行的触摸序列。例如当来电中断应用时。此时UIKit通过调用touchesCancelled(_:with:) 来通知您的视图。

参考资料：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_touches_in_your_view

关于UIKit里的手势

USE GESTURE RECOGNIZERS TO SIMPLIFY TOUCH HANDLING

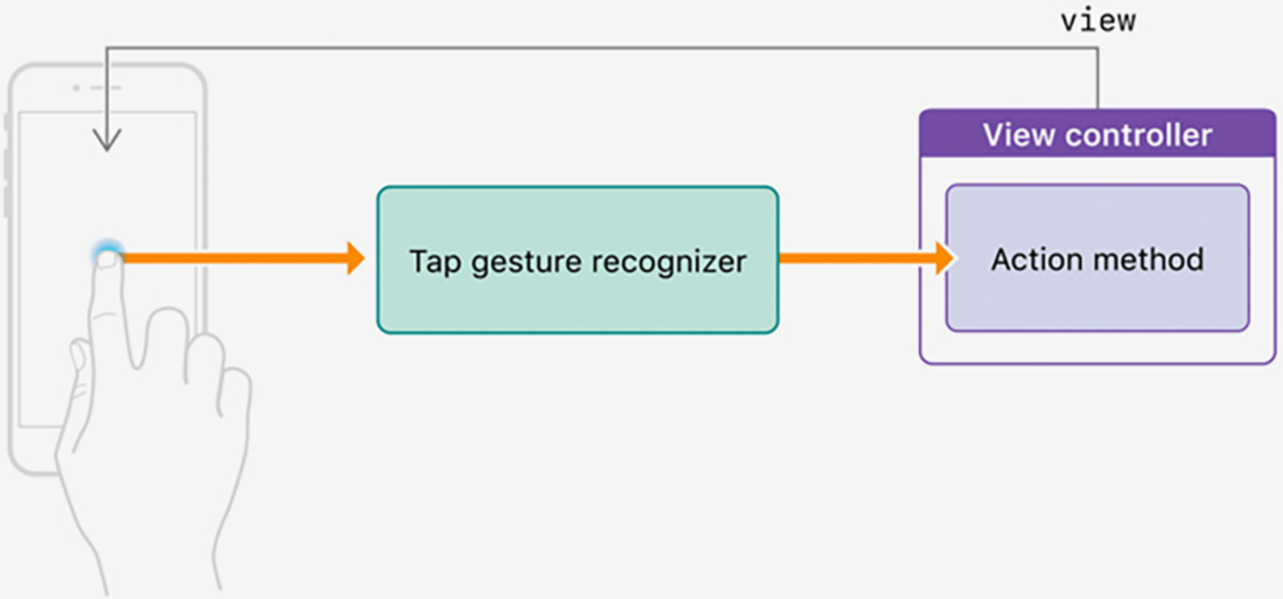
手势识别器是处理视图中的触摸或新闻事件最简单的方法。

您可以将一个或多个手势识别器附加到任何视图。

手势识别器封装处理和解释该视图传入事件所需的所有逻辑，并将它们与已知模式匹配。

检测到匹配时，手势识别器会通知其分配的目标对象，该对象可以是视图控制器、视图本身或应用中的任何其他对象。

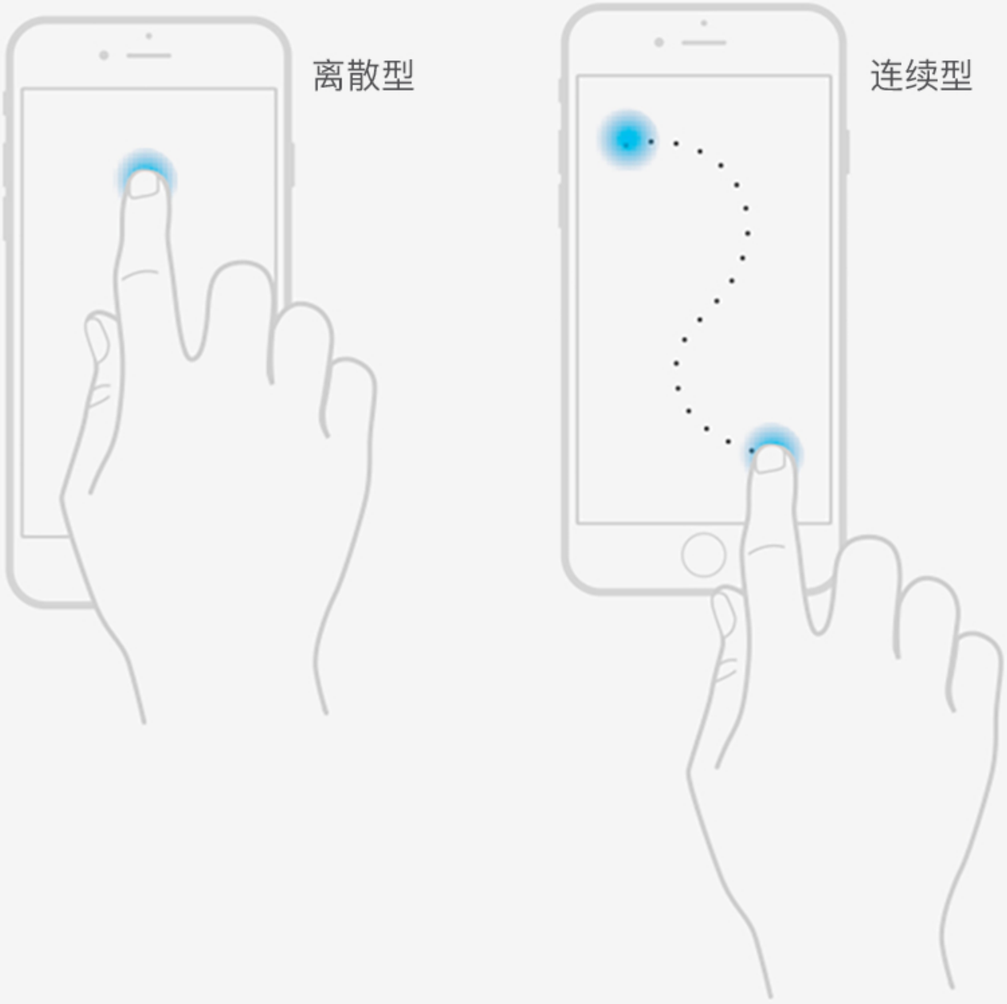
手势识别器使用目标–操作(target–action设计)模式发送通知。当UITapGesture 识别器对象检测到视图中的单指点击时，它将调用视图控制器的操作方法，用于提供响应。



参考资料：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

两种类型的手势识别器

GESTURE RECOGNIZERS COME IN TWO TYPES: DISCRETE AND CONTINUOUS



手势识别器有两种类型：离散和连续。

离散手势识别器在识别手势后，完成调用您的操作方法一次。

满足初始识别条件后，连续手势识别器会多次执行您的操作方法，每当手势事件中的信息发生更改时，都会通知您。

例如，每次触摸位置更改时，UIPanGestureRecognizer识别器对象都会调用您的操作方法。

参考资料：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

六种不同功能的手势

Gesture recognizers come in two types: discrete and continuous

Tap
(~0.1 second)

单击手势可短暂检测一个或多个手指触摸屏幕。这些手势中涉及的手指不得从初始触摸点显著移动，并且您可以配置手指必须触摸屏幕的次数。

例如，您可以配置点击手势识别器来检测单次点击、双击或三次点击。

如果未调用点击手势识别器的代码，请检查以下条件是否为真，并根据需要进行更正：

1、视图的 isUserInteractionEnabled 交互启用属性是否设置为 true。默认情况下，图像视图和标签将此属性设置为 false。

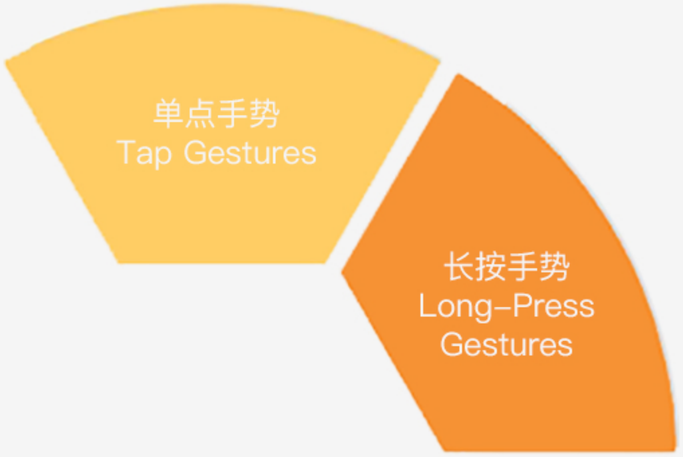
2、点击次数是否等于numberOfTapsRequired属性中指定的数字。

3、手指按下的数量是否等于 numberOfTouchesRequired 属性中指定的数字。

资源来源：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

六种不同功能的手势

Gesture recognizers come in two types: discrete and continuous



Long press
(>0.5 seconds)

1、长按（也称为按住）手势可长时间检测一个或多个手指（或手写笔）触摸屏幕。您可以配置识别按压所需的最短持续时间，以及手指必须触摸屏幕的次数。

2、手势识别器仅由触摸持续时间触发，而不是由与其关联的力度触发。用户的手指触摸屏幕达到最短时间(0.5s)后，长按手势识别器将进入 UIGestudreRecognizer.State.began 状态。

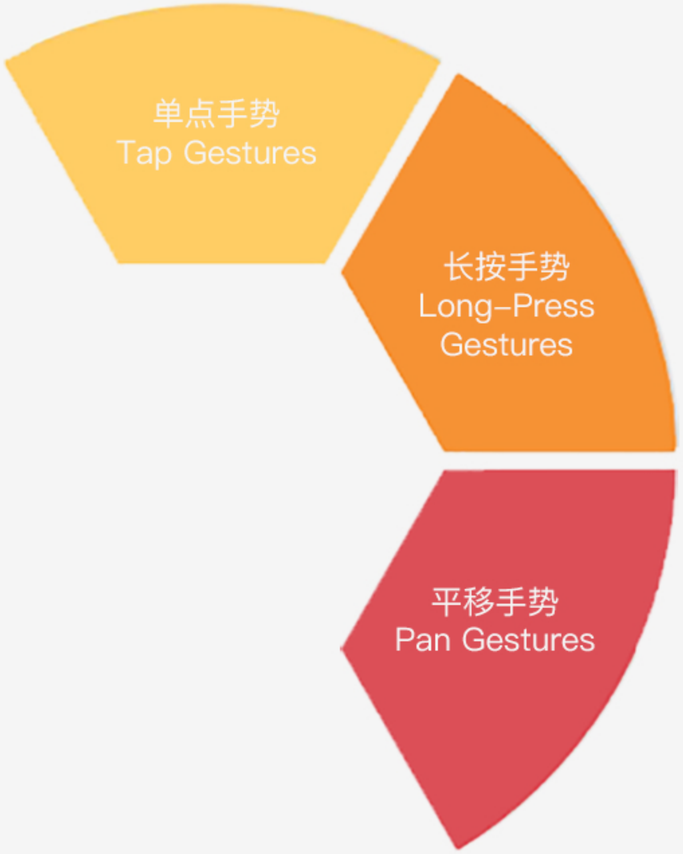
3、如果手指移动或触摸发生任何其他更改，识别器将切换到UIGestureRecognizer.State.changed。即使这些手指移动到初始视图之外，手势识别器仍保持 changed 状态.

4、当用户的手指从屏幕上抬起时，手势识别器将进入 UIGestureRecognizer.State.ended 状态。

资源来源：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

六种不同功能的手势

Gesture recognizers come in two types: discrete and continuous



每当用户在屏幕上移动一个或多个手指时，就会发生平移手势。

使用 UIPanGestureRecognizer 类检测平移手势，使用 UIScreenEdgePanGestureRecognizer 类检测屏幕边缘平移手势。

您可以通过以下方式给视图添加手势识别器：

1、编程方式： 给视图发送 addGestureRecognizer(_:)消息

2、故事板方式：从组件库中拖动平移手势，并将其拖放到视图中。

资源来源：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

六种不同功能的手势

Gesture recognizers come in two types: discrete and continuous

1、当用户以特定水平或垂直方向在屏幕上移动一个或多个手指时，将发生轻扫手势。使用 UISwipeGestureRecognizer 类检测轻扫手势。

2、UISwipeGestureRecognizer 对象在屏幕上水平或垂直跟踪用户手指的运动。

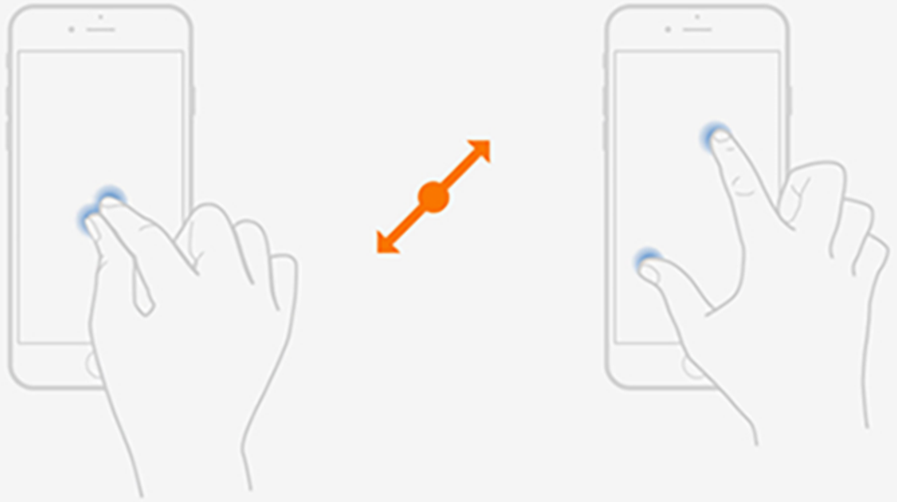
3、轻扫要求用户的手指向特定方向移动，并且不会明显偏离主要移动方向。手势所需的方向和手指数是可配置的。

4、轻扫手势是离散的，所以只有在手势成功结束后才会调用您的操作方法。因此，当您只关心手势的结果而不是跟踪用户手指的移动时，轻扫是最合适的。

资源来源：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

六种不同功能的手势

Gesture recognizers come in two types: discrete and continuous



1、捏合手势是一种连续的手势，用于跟踪接触屏幕的前两个手指的距离。

2、捏合手势是连续的，因此每次手指之间的距离发生变化时，都会调用您的操作方法。

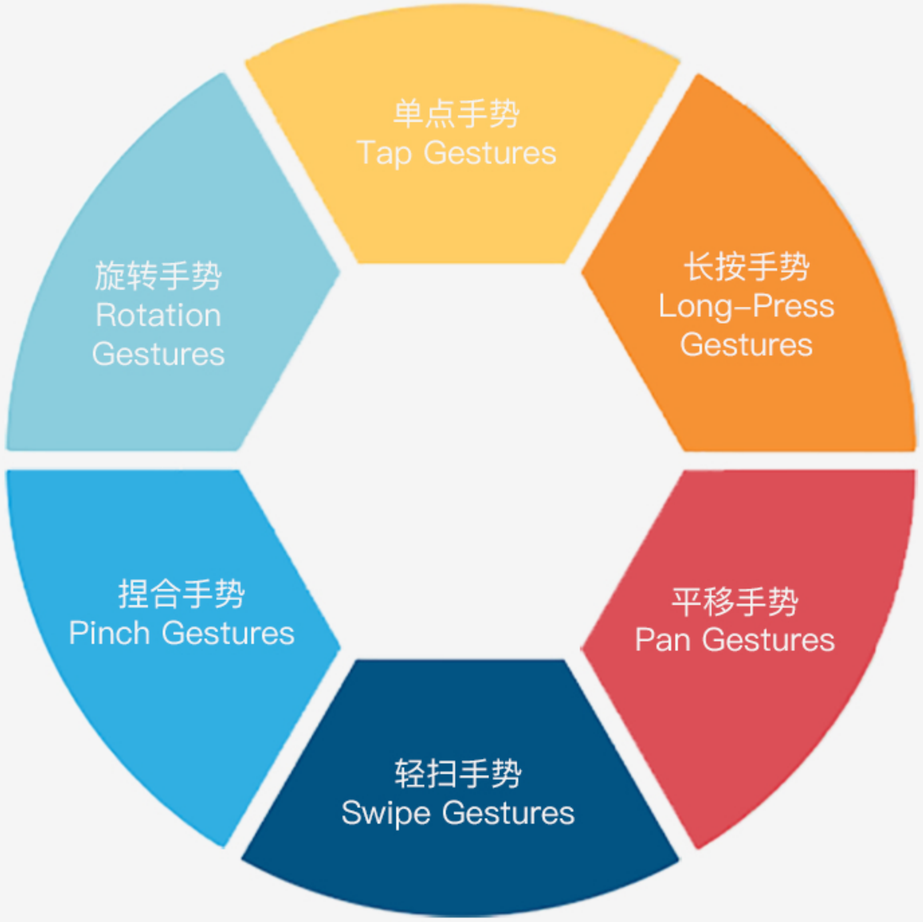
3、手指之间的距离报告为比例因子。在手势的开头，比例因子为 1.0。随着两个手指之间的距离增加，比例因子成比例增加。同样比例因子会随着手指的间距减小而减小。

4、捏合手势最常用于更改屏幕上的对象或内容的大小。例如，地图视图使用捏合手势来更改地图的缩放级别。

资源来源：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

六种不同功能的手势

Gesture recognizers come in two types: discrete and continuous



1、旋转手势是当接触屏幕的前两根手指相互旋转时发生的连续手势。您可以使用它们旋转视图或更新自定义控件的值。

2、旋转手势是连续的，因此每当旋转值发生更改时，都会调用您的操作方法，从而让您有机会更新内容。

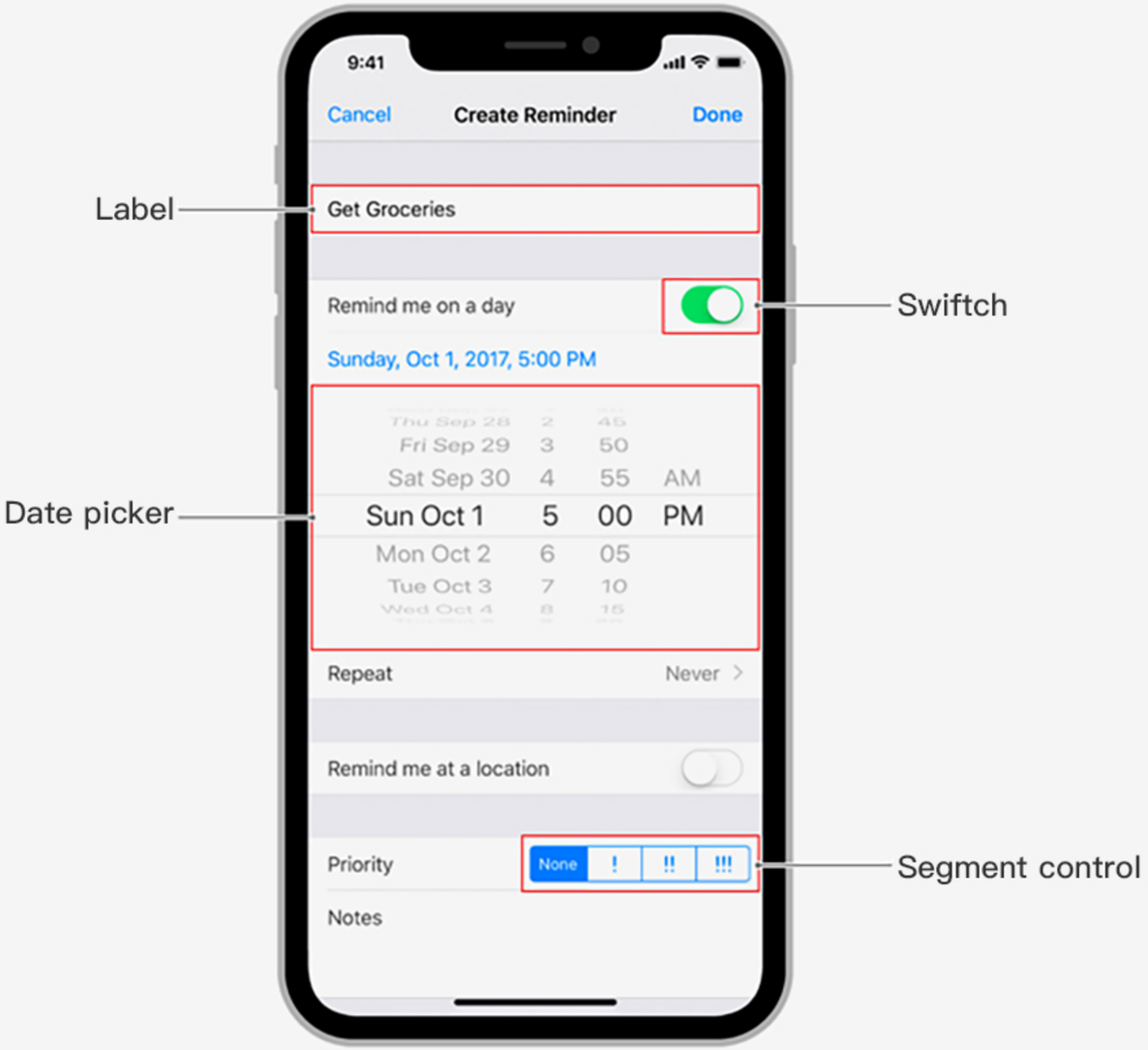
3、手势识别器以弧度报告旋转值。想象用户手指之间有一条线，手指在其初始位置创建的线表示测量的初始点，因此表示旋转角度为 0。

4、当手指移动时，在每个新位置的手指之间会创建一条新线。手势识别器测量初始线和新线之间的角度，并将结果置于其旋转属性中。

资源来源：https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/handling_uikit_gestures

iOS应用界面中的视图和控件

VIEWS AND CONTROLS



视图和控件是应用用户界面的可视化构建基块。使用它们在屏幕上绘制和组织应用的内容。

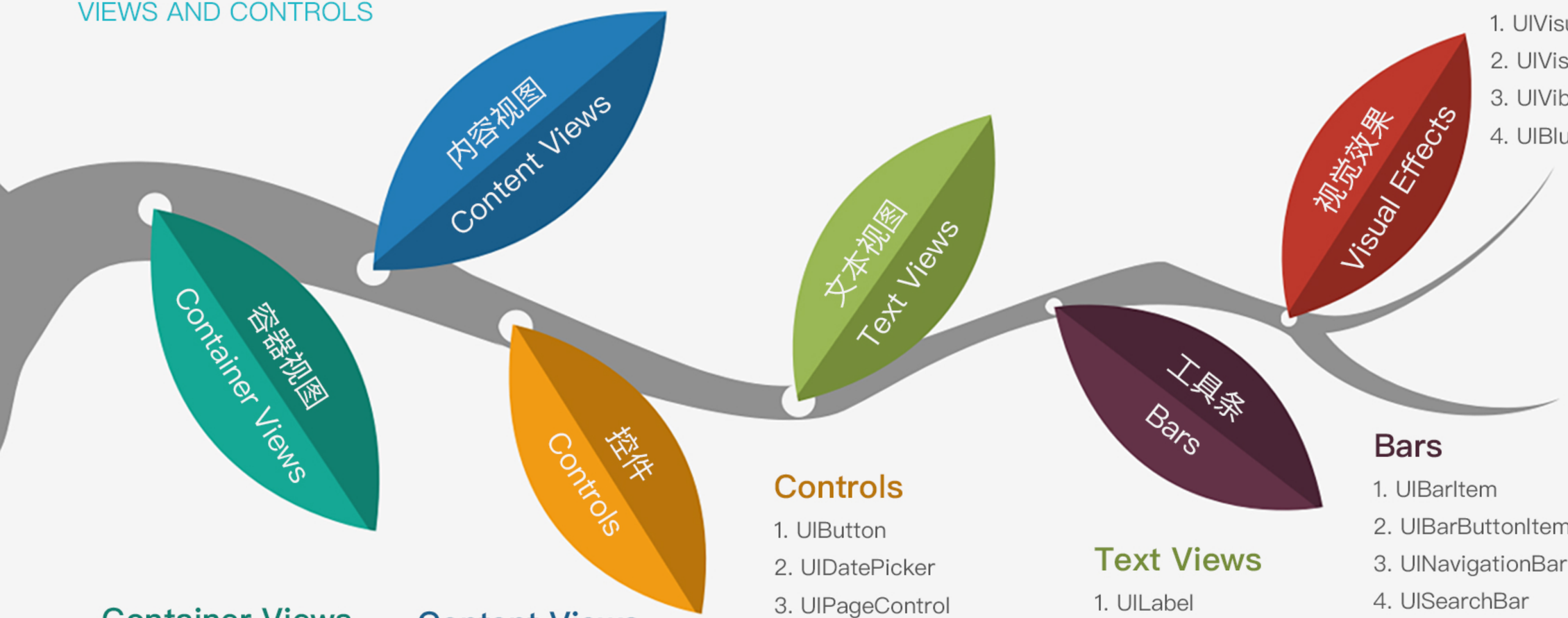
UIView 是所有视图的根类，并定义其常见行为。
UIControl 定义特定于按钮、开关和其他视图的行为，
这些行为是专为用户交互设计的。

视图可以承载其他视图。将一个视图嵌入另一个视图，会在主视图（父视图）和嵌入视图（子视图）之间创建包含关系。

参考资料：https://developer.apple.com/documentation/uikit/views_and_controls

iOS应用界面中的视图和控件

VIEWS AND CONTROLS



Container Views

1. Collection Views
2. Table Views
3. UIStackView
4. UIScrollView

Content Views

1. UIImageView
2. UIPickerView
3. UIActivityIndicatorView
4. UIProgressView

Controls

1. UIButton
2. UIDatePicker
3. UIPageControl
4. UISegmentedControl
5. UISlider
6. UIStepper
7. UISwitch

Text Views

1. UILabel
2. UITextField
3. UITextView
4. UISearchTextField

Visual Effects

1. UIVisualEffect
2. UIVisualEffectView
3. UIVibrancyEffect
4. UIBlurEffect

Bars

1. UIBarItem
2. UIBarButtonItem
3. UINavigationBar
4. UISearchBar
5. UIToolbar
6. UITabBar
7. UITabBarItem

参考资料：https://developer.apple.com/documentation/uikit/views_and_controls

视图的主要职责

BECAUSE VIEW OBJECTS ARE THE MAIN WAY YOUR APPLICATION INTERACTS WITH THE USER, THEY HAVE A NUMBER OF RESPONSIBILITIES.

布局和子视图管理

1、视图可能包含零个或多个子视图。

2、视图可以调整子视图的大小和位置。

3、使用"自动布局"调整视图大小和位置，以响应视图层次结构中的变化。

事件处理

1、视图是 UIResponder 的子类，可以响应触摸和其他类型的事件。

2、视图可以使用手势识别器来处理常见手势。

绘制内容和动画

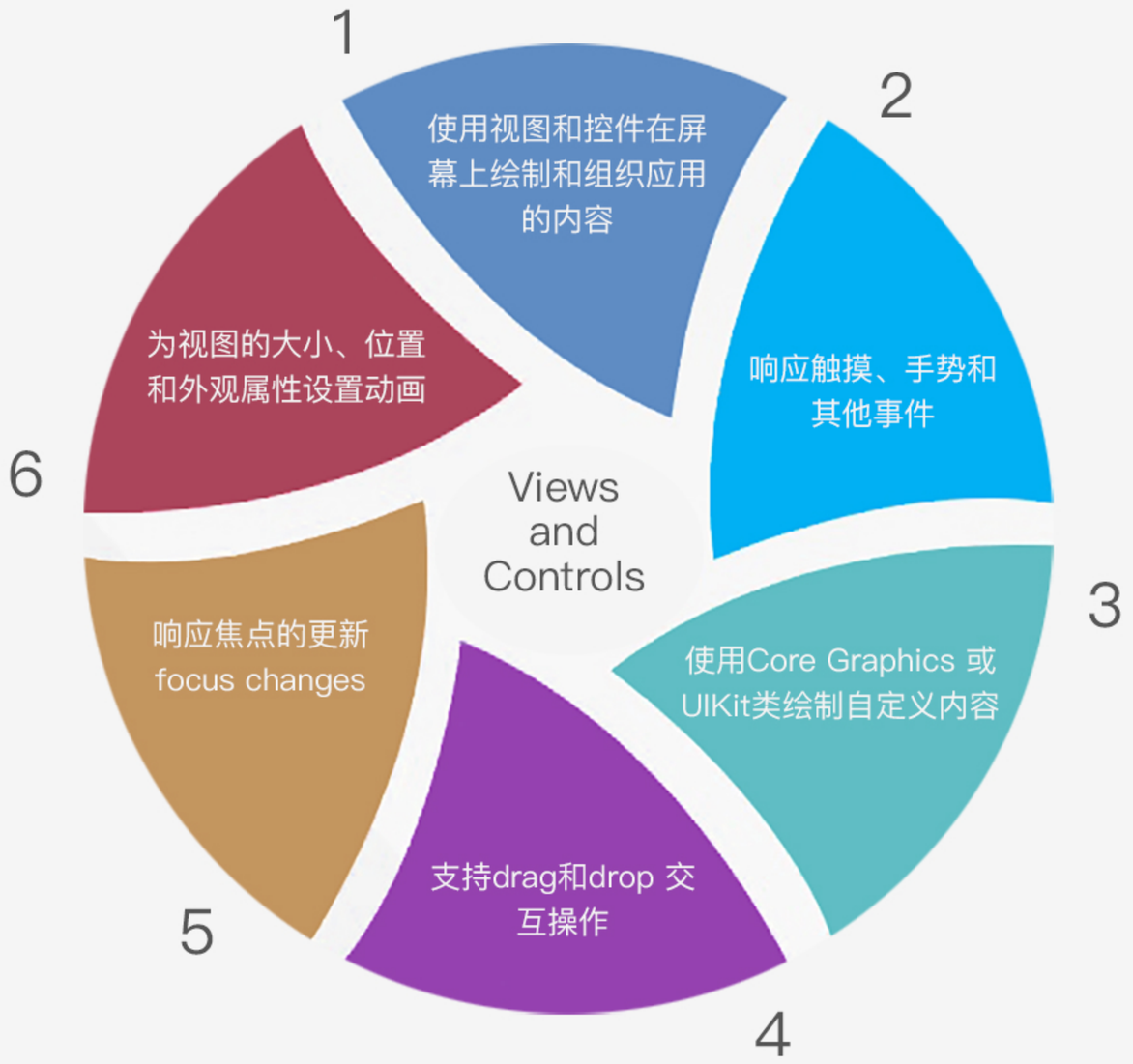
1、视图使用 UIKit 或 Core Graphics 在其矩形区域中绘制内容。

2、某些视图属性可以动画化为新值。

参考资料：https://developer.apple.com/documentation/uikit/views_and_controls

视图和控件的六种用途

YOU CAN USE VIEWS TO DO ANY OF THE FOLLOWING:



参考资料：https://developer.apple.com/documentation/uikit/views_and_controls

视图的层级关系

调整视图层级顺序的四种方式

addSubview(_:)方法

对 addSubview(_:) 方法 的每次调用，新视图都将置于所有其他同级视图的上方。

insertSubview(_:aboveSubview:) 方法

调用insertSubview(_:aboveSubview:) 方法，可以将新视图，放在指定同级视图的上方

insertSubview(_:belowSubview:) 方法

调用insertSubview(_:belowSubview:) 方法，可以将新视图，放在指定同级视图的下方

exchangeSubview(at:withSubviewAt:) 方法

调用exchangeSubview(at:withSubviewAt:) 方法，可以交换两个同级视图的层次。

参考资料：https://developer.apple.com/documentation/uikit/uiview

视图的六个动画属性

THE FOLLOWING PROPERTIES OF THE UIVIEW CLASS ARE ANIMATABLE:

frame属性表示视图在其父视图坐标系中的位置和大小。

bounds属性表示视图在其自己的坐标系里的位置和大小。

center属性表示视图框架矩形的中心点。

transform属性表示应用于视图的变换，该变换相对于视图边界的中心位置。

alpha属性表示视图的透明度。

backgroundColor属性表示视图的背景颜色。

参考资料：https://developer.apple.com/documentation/uikit/uiview

UIView视图和CALayer层

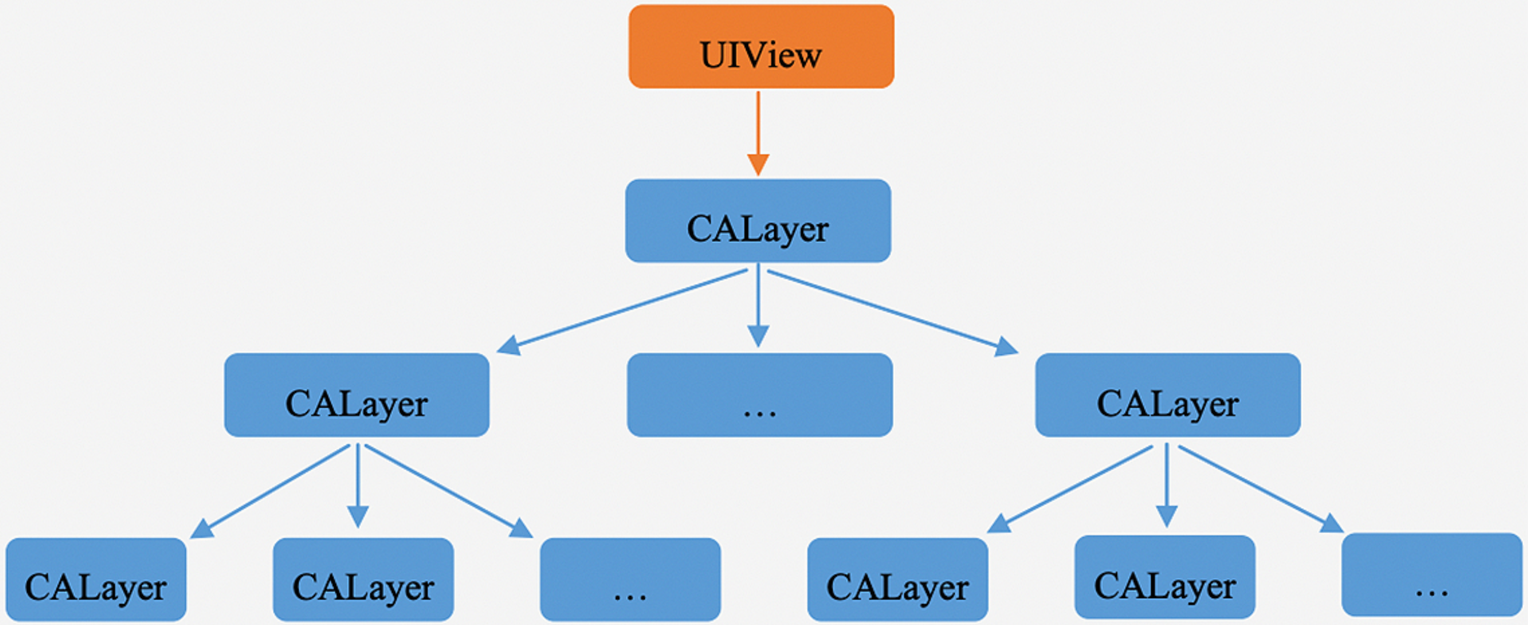
UIVIEW AND CALAYER

1、UIView是iOS系统中界面元素的基础，所有的界面元素都继承自它。

2、而UIView真正的绘图部分，是由一个叫CALayer(Core Animation Layer)的类来管理的。

3、UIView本身更像是一个CALayer的管理器，访问它的跟绘图和跟坐标有关的属性，例如frame，bounds等，实际上都是在内部访问它所包含的CALayer的相关属性。

4、一个Layer可以包含更多的子Layer，以创建复杂的视觉内容。



视图控制器加载视图的步骤

当在屏幕上显示视图控制器时，UIKit 必须首先加载并配置相应的视图，它使用以下步骤加载这些视图：

01 使用视图的 init(coder:) 方法创建每个视图

02 将视图连接到视图控制器中的相应actions 和 outlets

03 调用每个视图和视图控制器的awakeFromNib()方法

04 将视图层次结构分配给视图控制器的根视图

05 最后调用视图控制器的viewDidLoad() 方法

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/displaying_and_managing_views_with_a_view_controller

视图控制器加载视图的步骤

当视图即将显示时，UIKit会通知视图控制器，并通过以下步骤，更新这些视图的布局：

1、根据目标窗口更新视图控制器的特征集合。

2、调用viewWillAppear(_:)方法，让您知道视图控制器的视图即将出现在屏幕上。

3、如果需要，UIKit将更新当前布局边距，并调用viewLayoutMarginsDidChange方法。

4、如果需要，UIKit会更新安全区域内间距，并调用viewSafeAreaInsetsDidChange方法。

5、调用viewWillLayoutSubviews方法。

6、更新视图层次结构的布局。

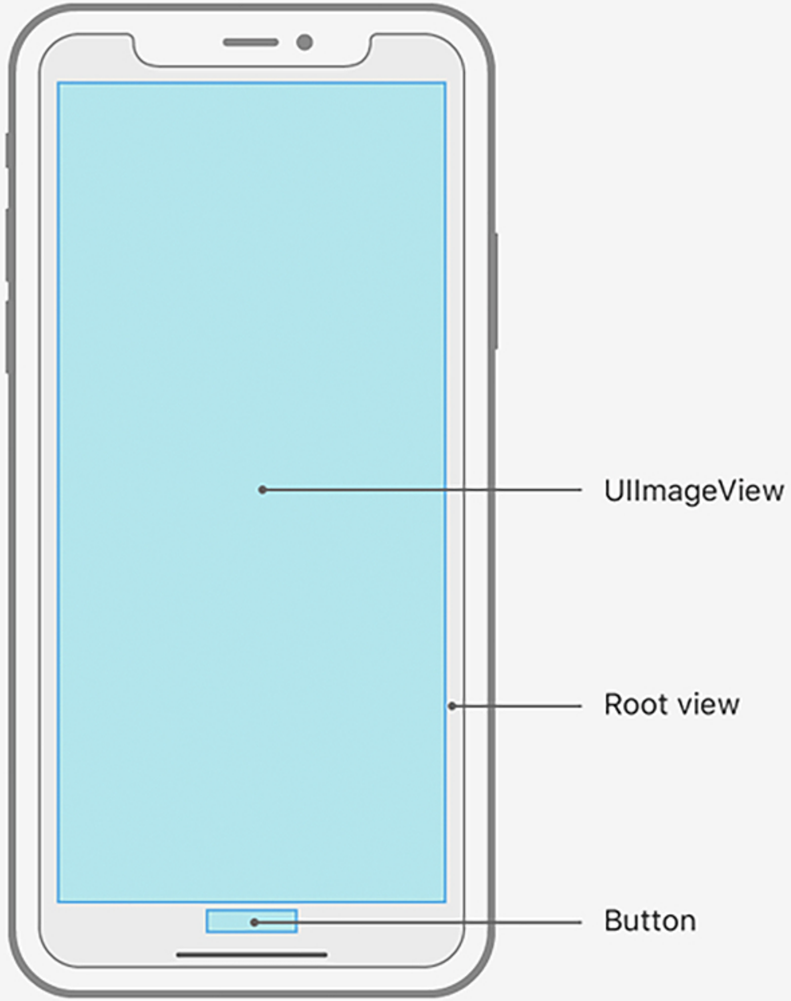
7、调用viewDidLayoutSubviews方法。

8、在屏幕上显示视图。

9、最后调用调用视图控制器的view-DidAppear(_:)方法。

向视图控制器添加视图

ADD VIEWS TO YOUR VIEW CONTROLLER



1、UIViewController 默认包含一个内容视图，可从视图属性访问(controller.view)，该视图属性作为控制器的视图层次结构的根视图。

2、对于该根视图，可以添加显示界面所需的任意数量的自定义视图。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/displaying_and_managing_views_with_a_view_controller

使用视图控制器搭建UI界面和连接数据

DEFINE VIEW CONTROLLERS FOR EACH UNIQUE PAGE OF CONTENT

表格界面

网格界面

表单界面

1、为新应用设计 UI 时，首先将该 UI 划分为不同的内容页面。

2、例如一个页面可能在表格中显示数据行。

3、另一个页面可能在网格中显示图像。

4、还有一个页面可能显示一组用于获取用户数据的表单字段。

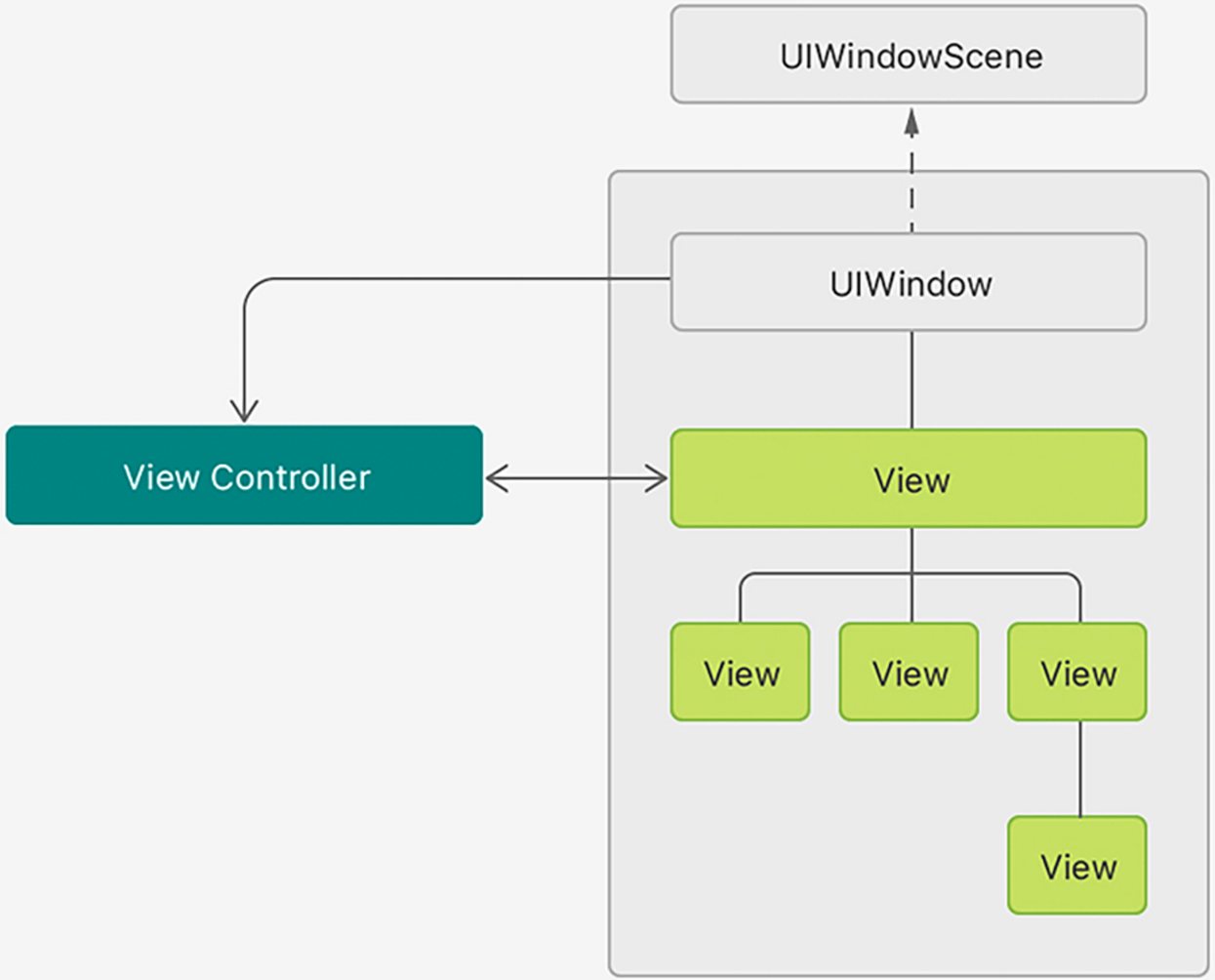
5、只有页面的结构和整体外观才是重要的，您从每个页面显示的特定数据并不重要。

6、事实上，许多应用在多个页面使用相同的布局，然后用不同的数据集填充每个副本。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/managing_content_in_your_app_s_windows

使用视图控制器有效管理视图

TO MANAGE YOUR VIEWS EFFECTIVELY WITH VIEW CONTROLLERS



1、应用界面中的每个场景，都包含一个窗口对象，以及一个或多个视图对象。

2、为了有效地管理视图，并便于从窗口中添加和删除视图，UIKit 提供了视图控制器。

3、视图控制器管理应用的单个视图集，并使这些视图中的信息保持最新。

4、每个窗口都有一个根视图控制器，用于指定窗口的初始视图集。

5、如果需要更改根视图控制器，需要给 UIKit 提供其他的视图控制器。

6、UIKit 处理从一组视图集到另一组视图集的转换，并通过多个视图控制器对象，管理应用的整个接口。因此，视图控制器在实现 UI 时，起着极为重要的作用。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/managing_content_in_your_app_s_windows

每个视图控制器都要做的事

EVERY VIEW CONTROLLER NEEDS TO DO THE FOLLOWING:

对于每个不同的页面，定义一个视图控制器来表示该页，
并管理其相应的视图。
每个视图控制器都需要执行以下操作：

01 使用数据填充视图，并在数据更改时更新这些视图。

02 当用户界面里的数据发生变化时，及时反向刷新数据模型对象。

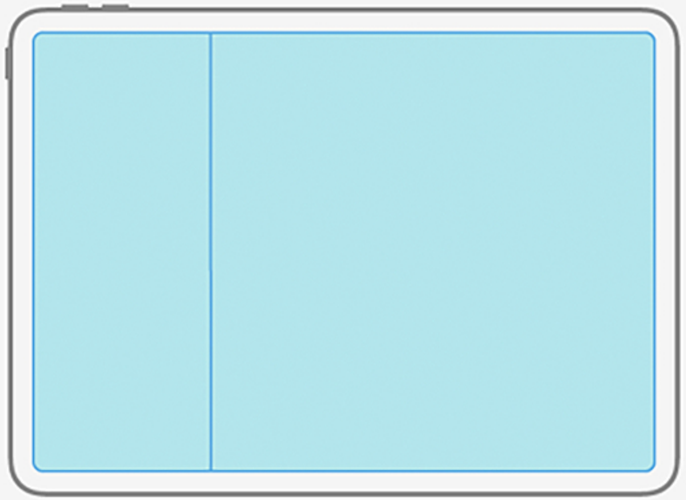
03 调整视图的大小、位置和可见性，以匹配当前的环境。

04 通过各种转场方式，过渡到其他的内容页面。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/managing_content_in_your_app_s_windows

为您的页面选择合适的导航模式一

CHOOSE THE NAVIGATION MODEL FOR YOUR CONTENT



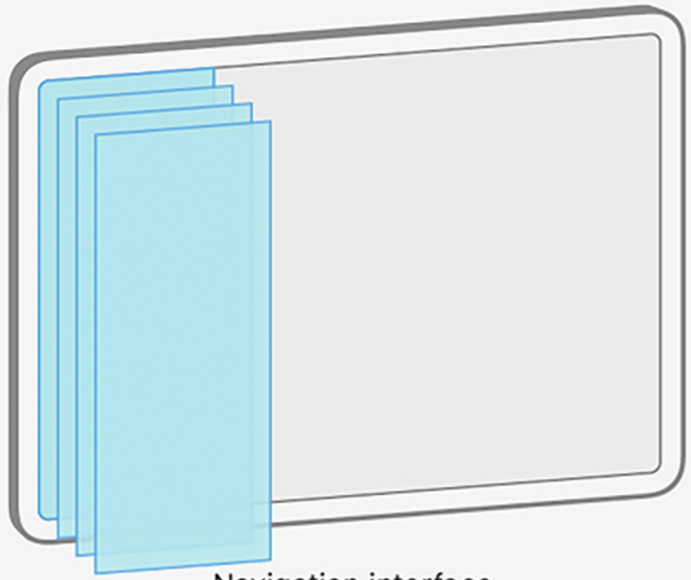
Split-view interface

UISplitViewController

1、分隔视图控制器在分隔视图界面中，并行管理两个子视图控制器（当空间可用时）。

2、当空间受限时，系统会在导航界面中显示这些视图控制器。

3、这种类型的界面也称为主要/详细信息界面。



Navigation interface

UINavigationController

1、导航控制器可以在导航界面中，管理多个子控制器形成的堆栈。

2、将新视图控制器推送Push到堆栈，可以替换当前的视图控制器。

3、弹出Pop视图控制器会显示上一个的视图控制器。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/managing_content_in_your_app_s_windows

为您的页面选择合适的导航模式二

CHOOSE THE NAVIGATION MODEL FOR YOUR CONTENT

Tab bar interface

Paged interface

UITabBarController

1、选项卡控制器在窗口底部显示选项卡，用于在不同模式之间进行选择和显示该模式的视图。

2、选项卡控制器的每个选项卡都与自定义视图控制器相关联。当用户选择特定选项卡时，选项卡控制器将显示相应视图控制器的根视图，替换之前的视图。

UIPageViewController

1、页面视图控制器是管理内容页之间导航的容器视图控制器，其中每个页面由子视图控制器管理。

2、页面视图控制器的导航可以通过编程方式控制，也可以使用手势直接控制。当在页面之间进行导航时，页面视图控制器使用指定的过渡效果。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/managing_content_in_your_app_s_windows

关于UINavigationController

CHOOSE THE NAVIGATION MODEL FOR YOUR CONTENT

导航控制器用于在导航界面中，管理一个或多个子视图控制器。在这种类型的接口中，一次只能看到一个子视图控制器。

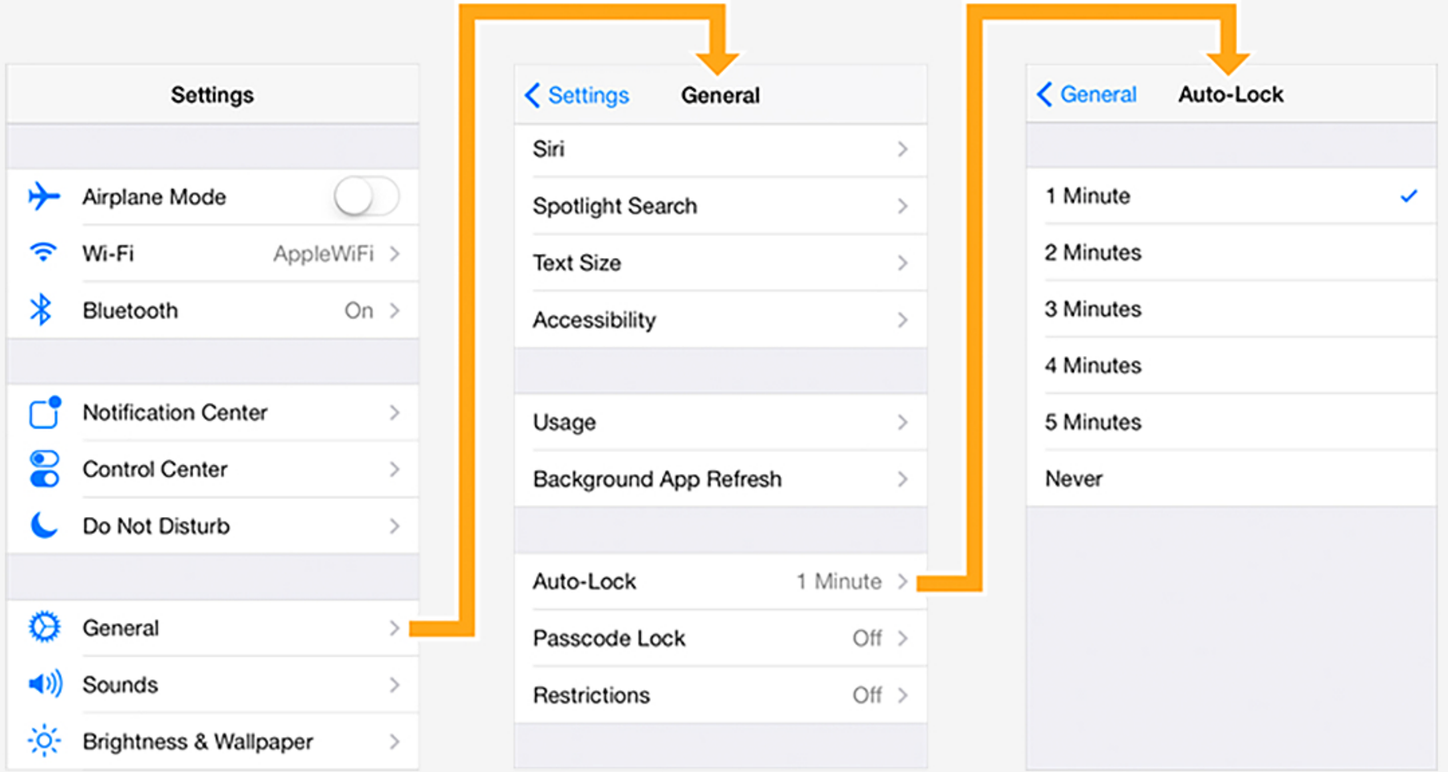
1、右图是 iOS 模拟器里的"设置"应用程序提供的导航界面示例。

2、第一个页面显示了包含首选项的应用程序列表。

3、选择General选项，将进入第二个页面，以显示和General相关的设置选项。

4、选择Auto-Lock选项，会显示更多设置。

5、对于除根视图外的所有视图，导航控制器提供一个后退按钮，允许用户向上移动层次结构。



A sample navigation interface

参考资料：https://developer.apple.com/documentation/uikit/uinavigationcontroller

UINavigationController的组成元素

OBJECTS MANAGED BY THE NAVIGATION CONTROLLER

1、导航控制器对象使用有序数组（导航堆栈）管理其子视图控制器。

2、数组中的第一个视图控制器是根视图控制器，表示堆栈的底部。

3、数组中的最后一个视图控制器是堆栈上最顶层的项，表示当前显示的视图控制器。

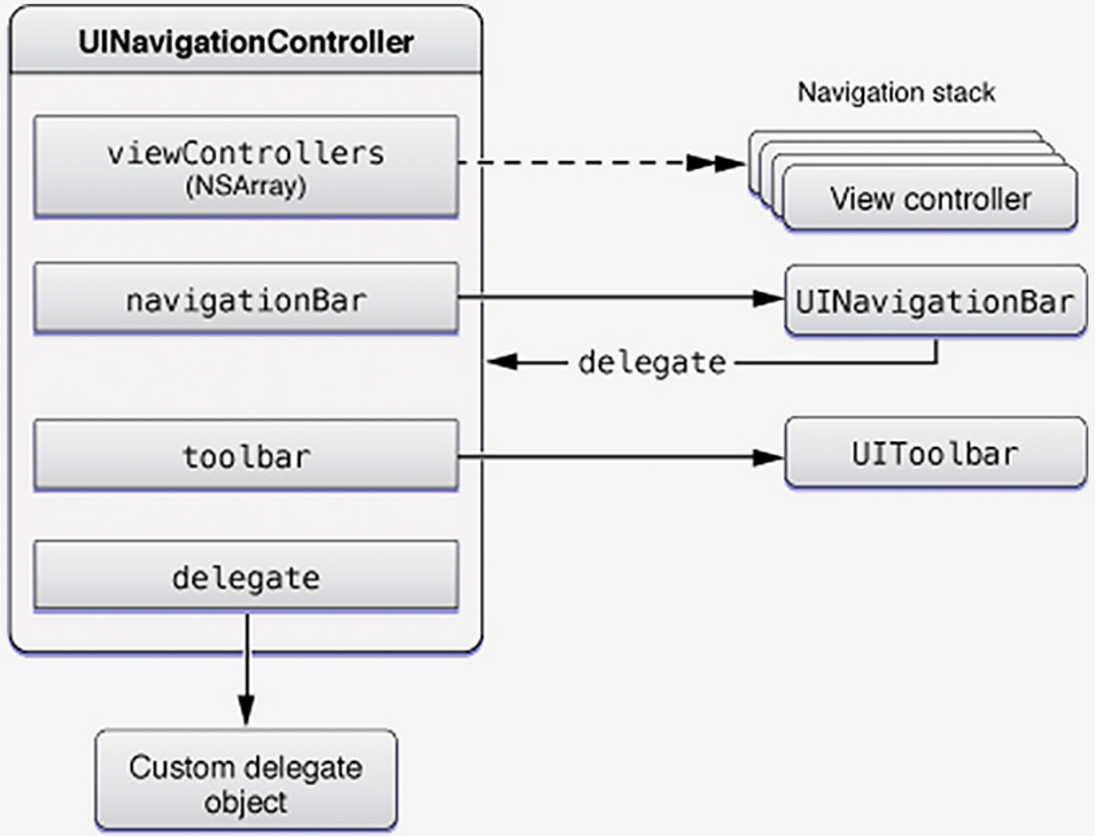
4、导航控制器管理界面顶部的导航栏和界面底部的可选工具栏。

5、导航栏始终存在，并由导航控制器本身管理，该控制器使用其子视图控制器提供的内容更新导航栏。

6、当 isToolbarHidden 属性为 false 时，导航控制器会使用最顶层视图控制器提供的内容更新工具栏。

6、导航控制器与其委托对象协调其自身的行为。

7、委托对象可以重载视图控制器的推送Push或弹出Pop，提供自定义的转场动画，并为导航界面指定首选方向。



参考资料：https://developer.apple.com/documentation/uikit/uinavigationcontroller

UINavigationController里的视图

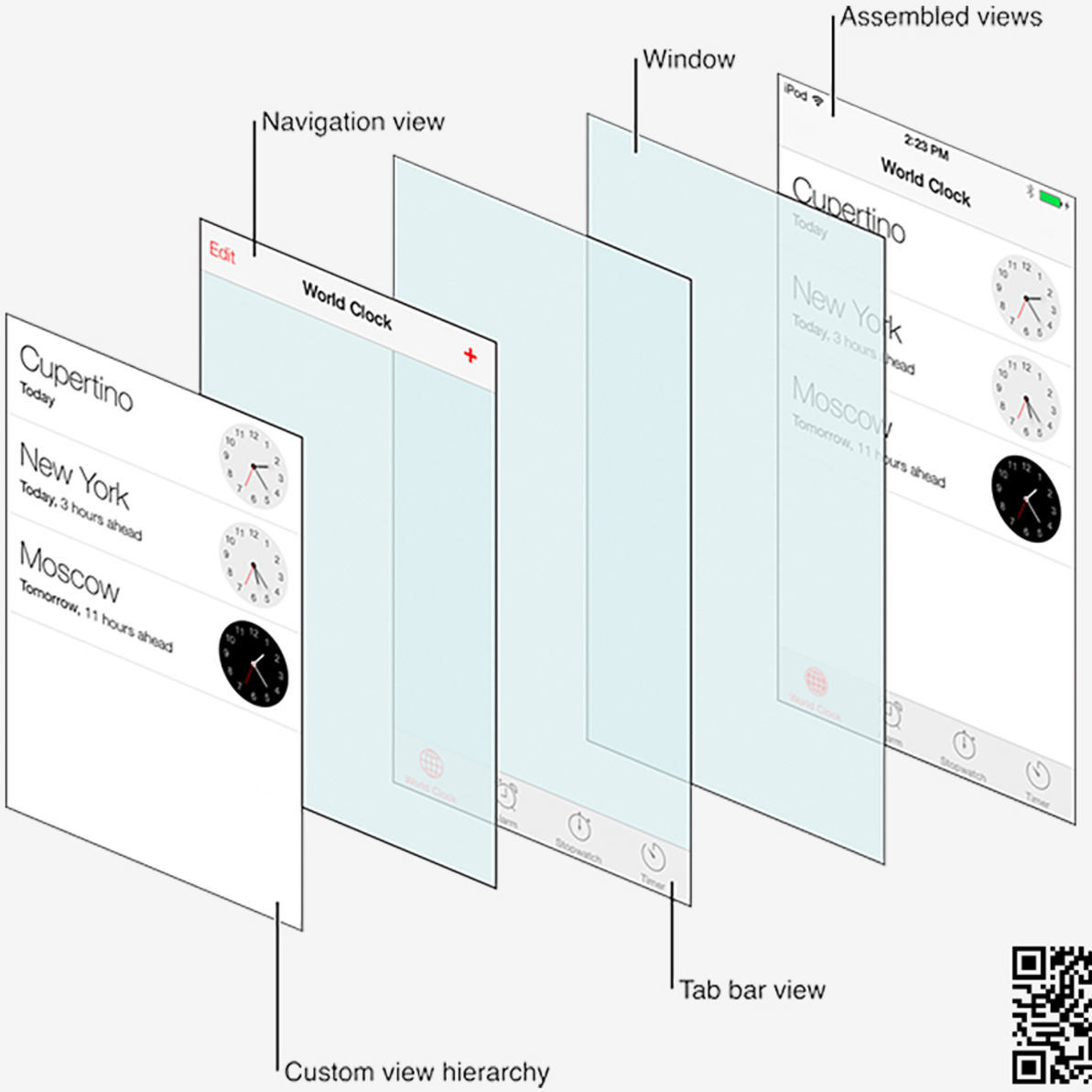
Views in the UINavigationController

1、导航控制器是一个容器视图控制器，也就是说，它将其他视图控制器的内容嵌入到自身内部。

2、从导航控制器的view属性，可以访问导航控制器的视图。此视图包含导航栏、可选工具栏和对应于最顶层视图控制器的内容视图。

3、在右图中，导航界面进一步嵌入在选项卡界面中。尽管导航栏和工具栏视图的内容有可能更改，但导航栏和工具栏本身不会更改。

4、唯一实际更改的视图，是导航堆栈上最顶层视图控制器提供的自定义内容视图。



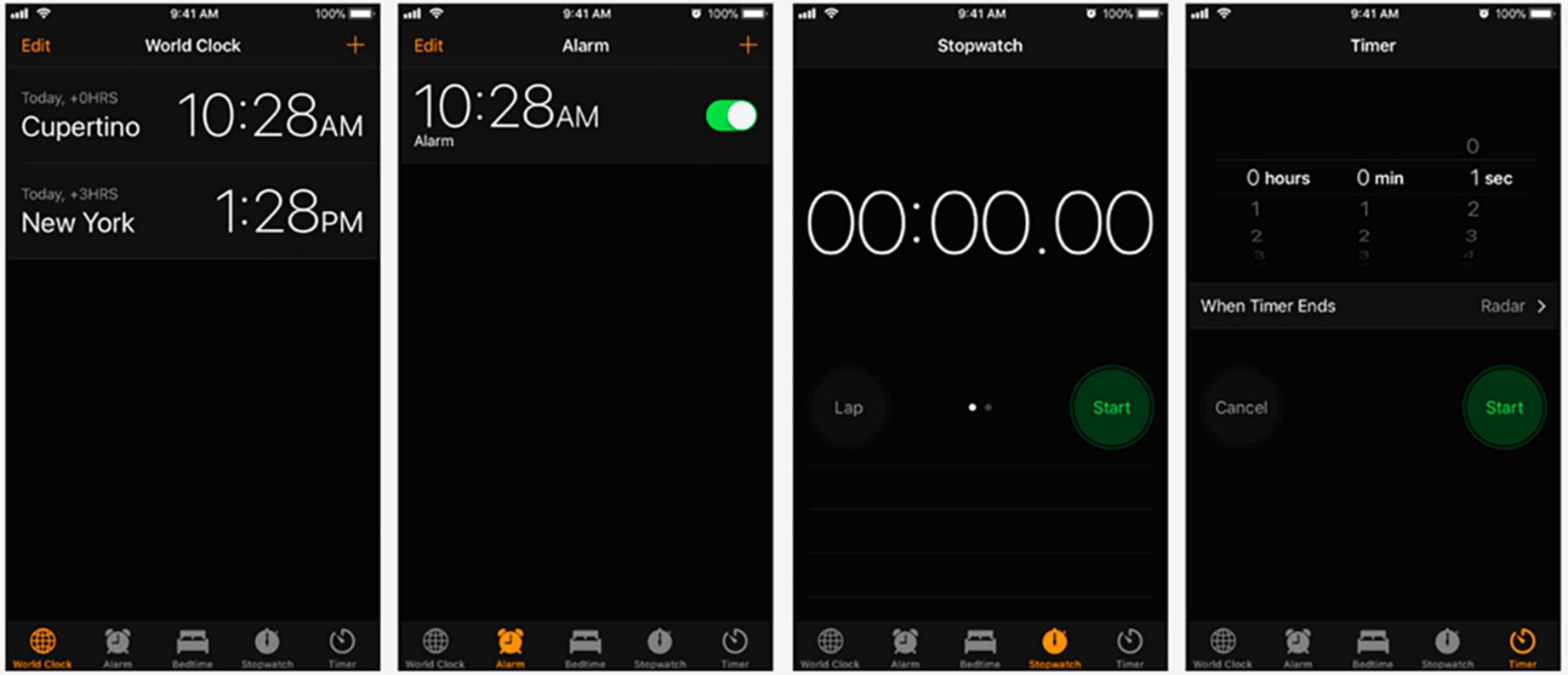
参考资料：https://developer.apple.com/documentation/uikit/uinavigationcontroller

关于UITabBarController

ABOUT THE UITABBARCONTROLLER

1、选项卡控制器在窗口的底部显示选项卡，用于在不同模式之间进行选择和显示该模式的页面。

2、选项卡控制器的每个选项卡都与自定义视图控制器相关联。当用户选择特定选项卡时，选项卡控制器将显示相应视图控制器的根视图，替换之前的任何视图。



参考资料：https://developer.apple.com/documentation/uikit/uitabbarcontroller

关于UITabBarController

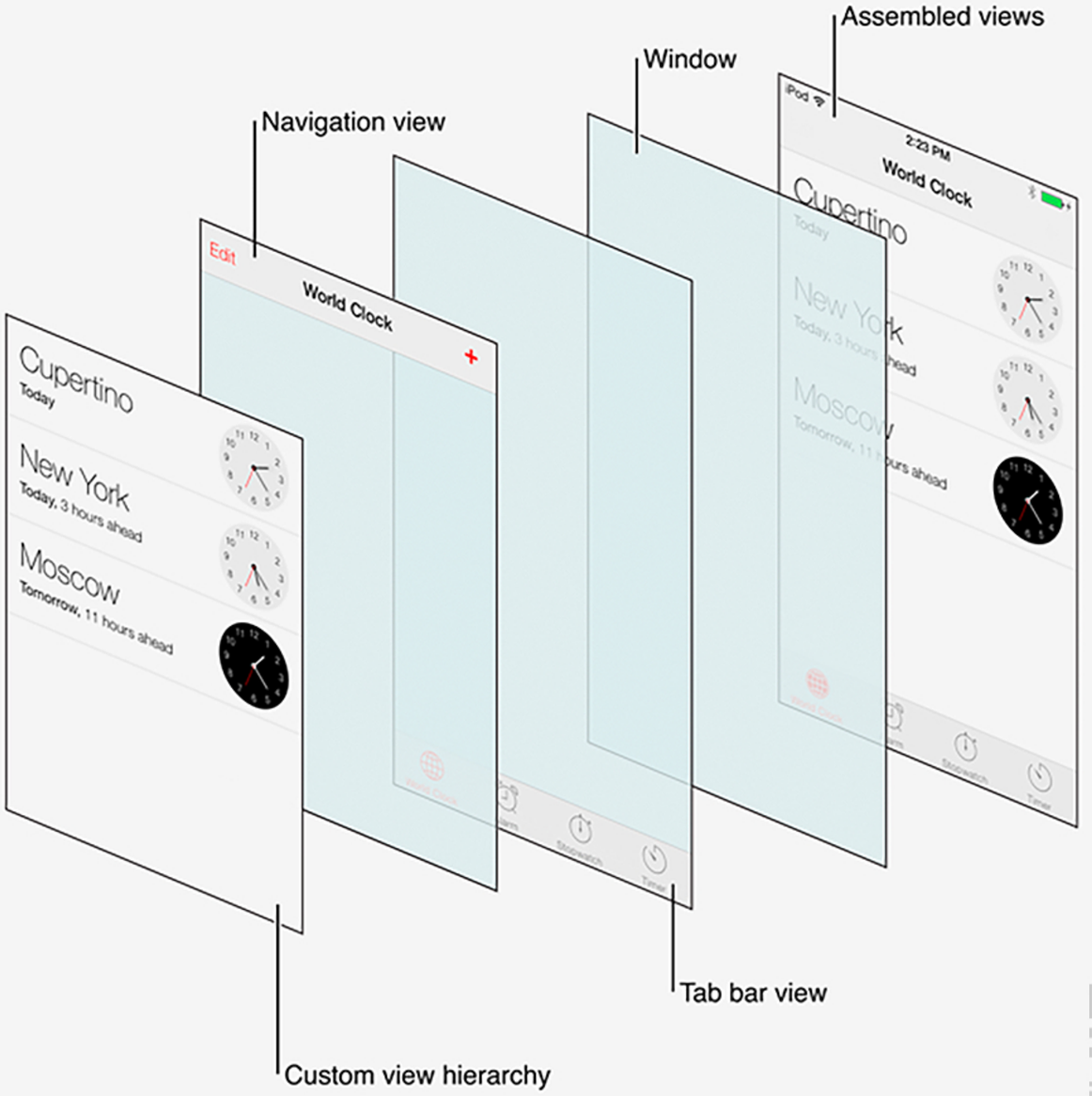
ABOUT THE UITABBARCONTROLLER

1、由于 UITabBarController 类继承自 UIViewController 类，因此选项卡控制器可以通过view属性访问自己的视图。

2、选项卡控制器的视图是一个容器，该窗口包含选项卡视图和包含自定义内容的视图。

3、选项卡视图为用户提供选择控件，由一个或多个选项卡项目组成。

4、您可以使用导航控制器或自定义视图控制器，作为选项卡的根视图控制器。如果根视图控制器是导航控制器，则选项卡控制器会进一步调整导航内容的大小，使其不与选项卡栏重叠。



参考资料：https://developer.apple.com/documentation/uikit/uitabbarcontroller

应用程序的MVC架构

About the Model–View–Controller

Controller是中介，协调Model和View相互协作

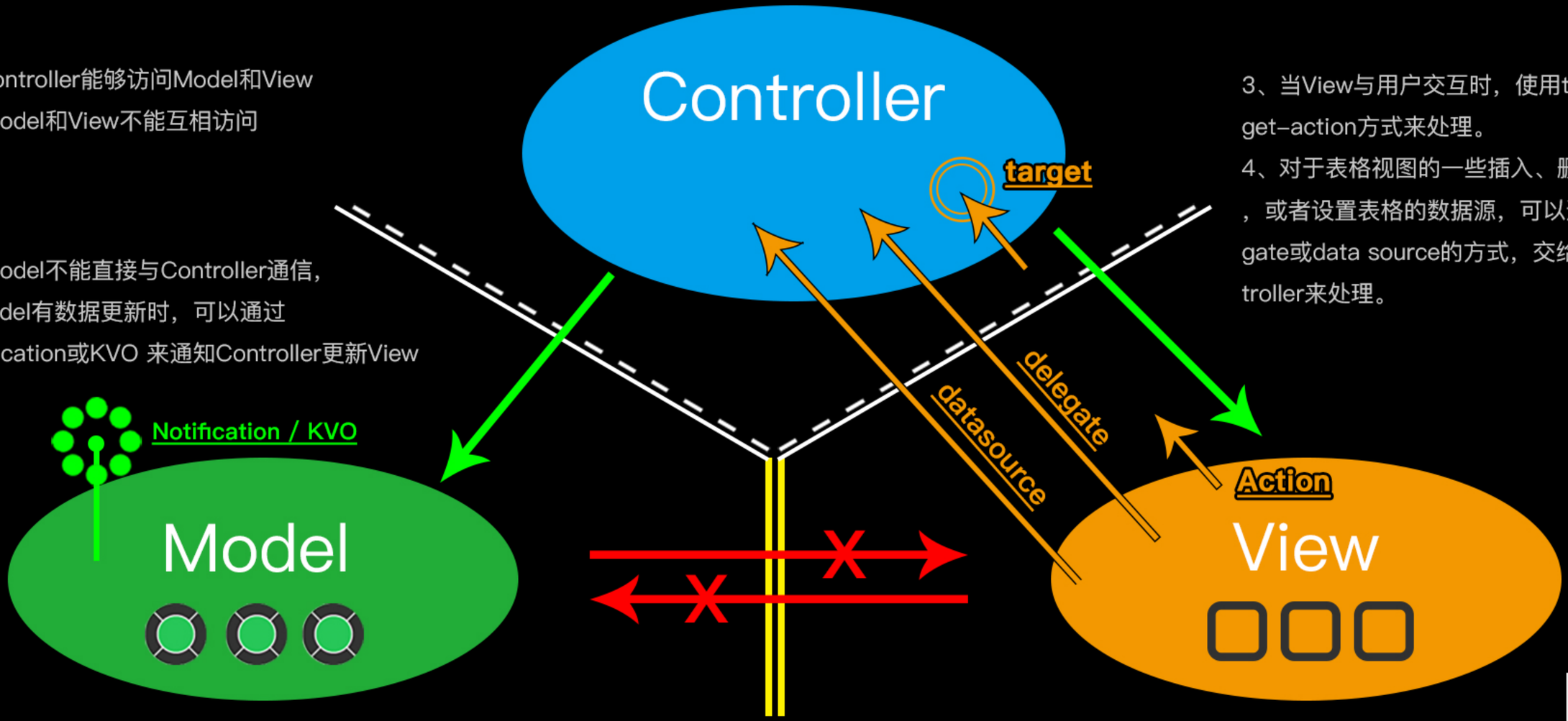
1、Controller能够访问Model和View

2、Model和View不能互相访问

3、当View与用户交互时，使用tar–get–action方式来处理。

4、对于表格视图的一些插入、删除操作，或者设置表格的数据源，可以通过dele–gate或data source的方式，交给Con–troller来处理。

5、Model不能直接与Controller通信，当Model有数据更新时，可以通过Notification或KVO 来通知Controller更新View

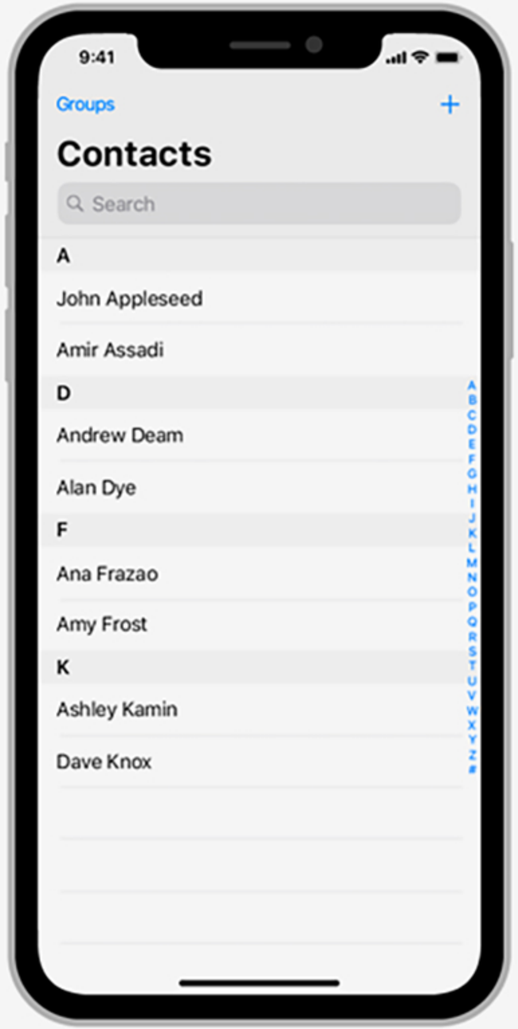


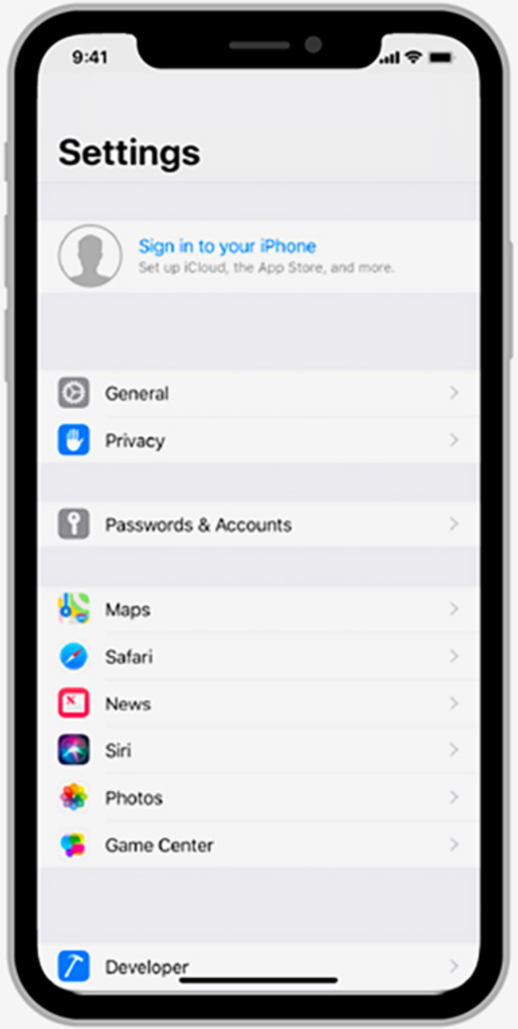
Model负责存储数据和处理业务逻辑

View负责显示数据和与用户交互

关于UITableView的使用

A view that presents data using rows arranged in a single column.





1、表视图显示一列垂直滚动内容，分为几行或几节。

2、表中的每一行都包含应用的一部分内容。例如"联系人"应用，在单独的行中，显示每个联系人的名称。"设置"应用的主页显示可用的设置组。

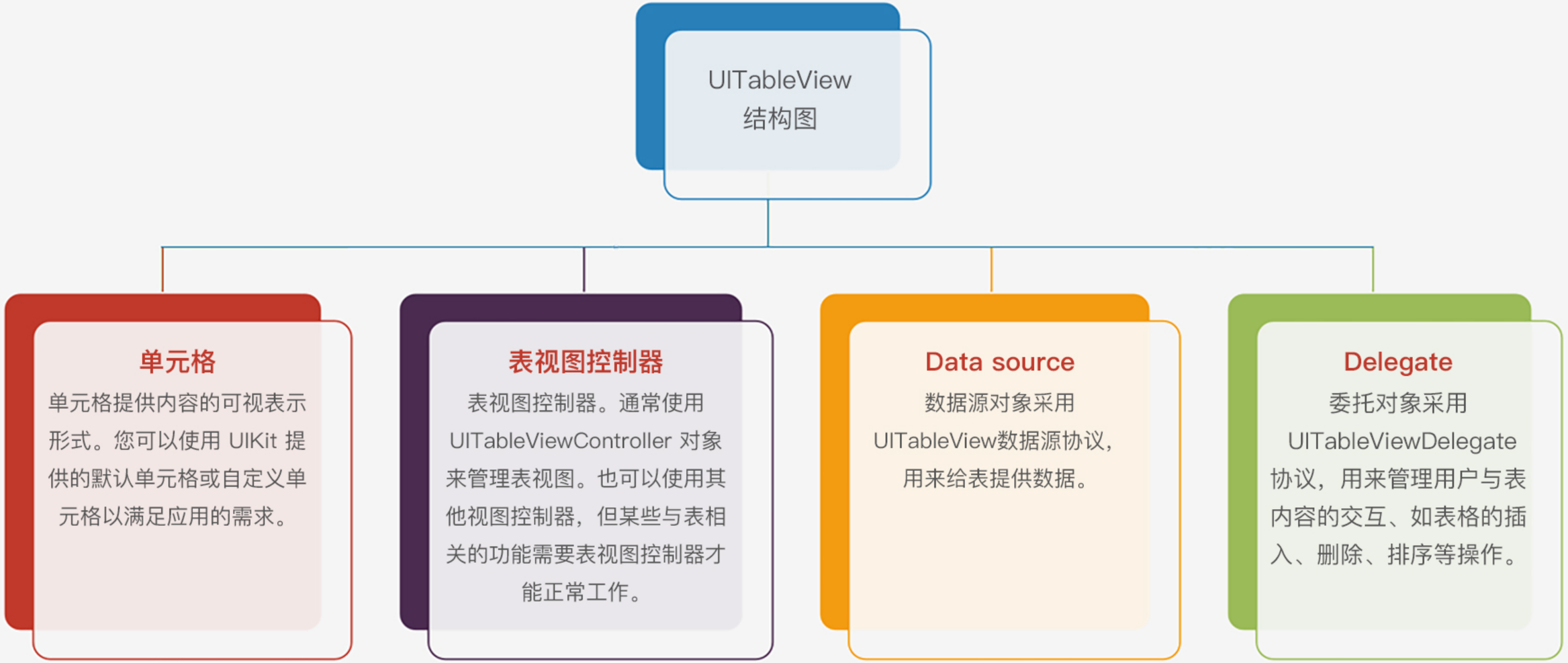
3、您可以将表配置为显示单个长行列表，也可以将相关行分组到节中，以便更轻松地导航内容。

4、表通常由数据高度结构化或分层组织的应用使用。包含分层数据的应用，通常将表与导航视图控制器结合使用，这便于在层次结构的不同级别之间导航。例如"设置"应用，它使用表和导航控制器来组织系统设置。

参考资料：https://developer.apple.com/documentation/uikit/uitableview

UITableView的组成部分

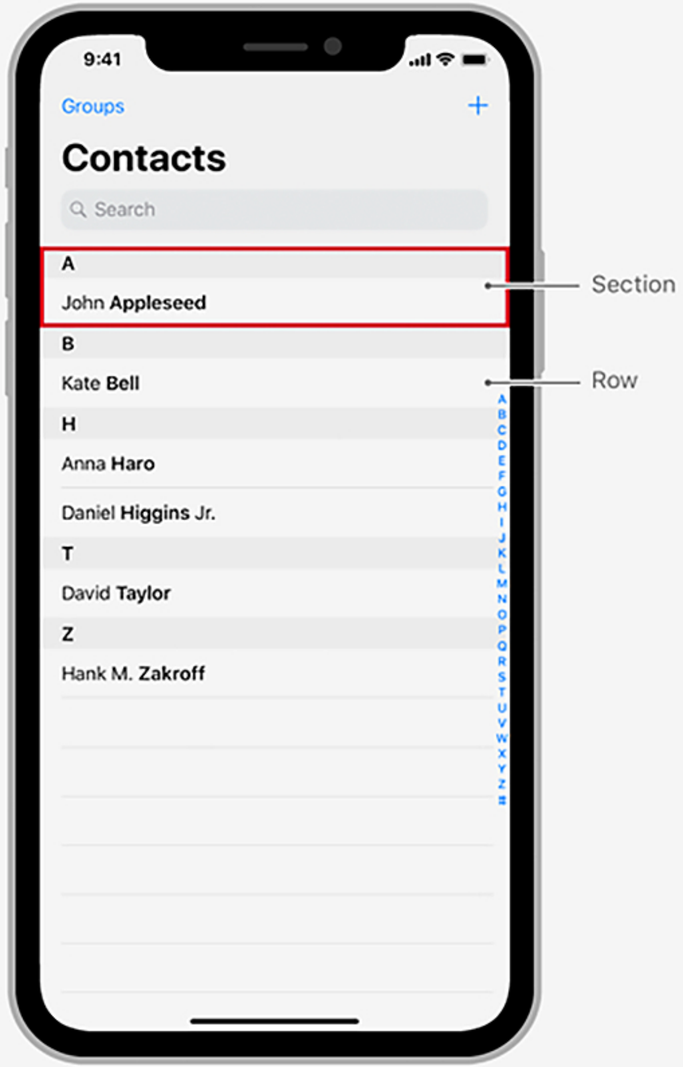
Table views are a collaboration between many different objects, including:



参考资料：https://developer.apple.com/documentation/uikit/views_and_controls/table_views

UITableView的组成部分

Table views are a collaboration between many different objects, including:



1、表视图将数据组织成行和节。

2、行显示单个数据项，并将相关行分组在一起。

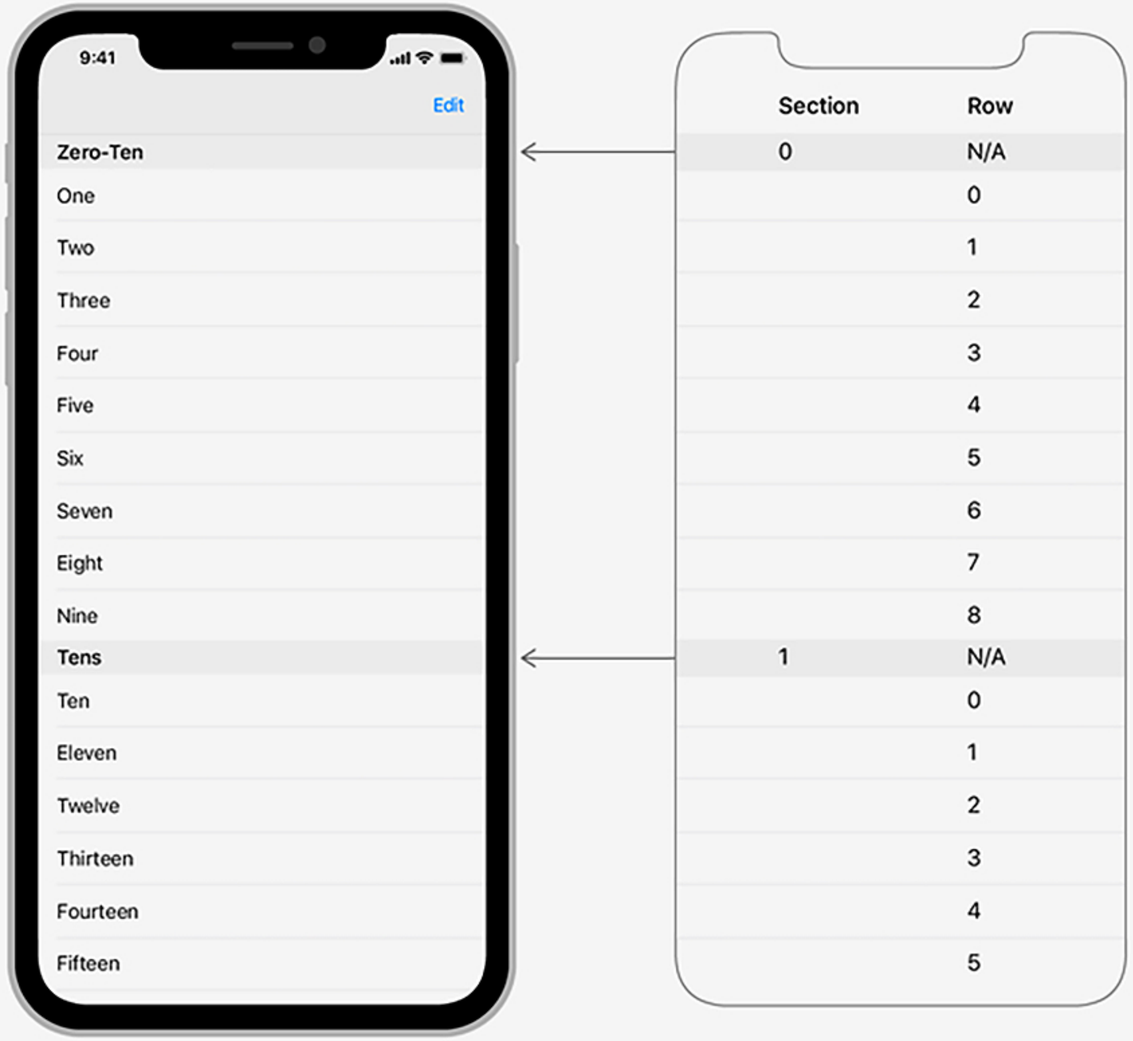
3、节不是必需的，但它们是组织分层数据的好方法。

4、例如"联系人"应用，它可以在每行显示联系人的名称，并根据联系人姓氏的第一个字母，将行分组到节中。

参考资料：https://developer.apple.com/documentation/uikit/views_and_controls/table_views

指定行和节的位置

Specifying the Location of Rows and Sections



1、表视图使用 IndexPath 对象的行和节属性，将单元格的位置传达给您。

2、行和节索引基于零，因此第一部分位于索引 0，第二部分位于索引 1，等等。

3、同样，每节的第一行在索引 0 处，这意味着您需要 IndexPath 对象的 section 值和 row 值来唯一地标识行。如果表没有 section ，则只需要 row 值即可。

4、例如第2节第3行的单元格的 indexPath 为：

indexPath(section: 1, row: 2)

参考资料：https://developer.apple.com/documentation/uikit/uitableviewdatasource

UITableViewDataSource的作用

To manage the data, you provide the table with a data source object

数据源对象响应表里的数据相关的请求。它还直接管理表的数据。

为表的每一行提供单元格。

设置表中的节数和行数。

提供节的标题和页脚的标题。

如果表格包含索引，数据源对象需要配置表的索引。

更新基础数据，以响应用户或表的更新。

参考资料：https://developer.apple.com/documentation/uikit/uitableviewdatasource

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源：［1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限］

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 使用中 使用中 使用中 使用中 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 使用中 使用中 使用中 使用中 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 未使用 | 使用中 | 使用中 | 使用中 | 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 未使用 | 使用中 | 使用中 | 使用中 | 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

6、由于手指是向上方滑动的，所以需要在紫色单元格的下方补充一个单元格。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 未使用 | 使用中 | 使用中 | 使用中 | 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

6、由于手指是向上方滑动的，所以需要在紫色单元格的下方补充一个单元格。

7、这时通过表格视图的dequeueReusableCell方法，从单元格列表里，获取状态为未使用的单元格，并将它放置在紫色单元格的下方。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 使用中 使用中 使用中 使用中 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

6、由于手指是向上方滑动的，所以需要在紫色单元格的下方补充一个单元格。

7、这时通过表格视图的dequeueReusableCell方法，从单元格列表里，获取状态为未使用的单元格，并将它放置在紫色单元格的下方。

8、同时更改单元格里的标签内容为6，并将它的状态设置为使用中。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 使用中 使用中 使用中 使用中 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

6、由于手指是向上方滑动的，所以需要在紫色单元格的下方补充一个单元格。

7、这时通过表格视图的dequeueReusableCell方法，从单元格列表里，获取状态为未使用的单元格，并将它放置在紫色单元格的下方。

8、同时更改单元格里的标签内容为6，并将它的状态设置为使用中。

9、红色单元格和其它的单元格，继续随着手指向上方滑动。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 使用中　未使用　使用中　使用中　使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

6、由于手指是向上方滑动的，所以需要在紫色单元格的下方补充一个单元格。

7、这时通过表格视图的dequeueReusableCell方法，从单元格列表里，获取状态为未使用的单元格，并将它放置在紫色单元格的下方。

8、同时更改单元格里的标签内容为6，并将它的状态设置为使用中。

9、红色单元格和其它的单元格，继续随着手指向上方滑动。

10、橙色单元格从顶部滑出屏幕，它的状态变为未使用。接着再次重复第4到9的步骤。

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

以动画的方式演示单元格的复用机制

To manage the data, you provide the table with a data source object

表格的数据源： [1, 2, 3, 4, 5, 6, 7, 8, 9, ... 无限]

表格的单元格：

单元格的状态： 使用中 使用中 使用中 使用中 使用中

1、右侧的表格用来显示一列数字，数字的范围为1到无限。

2、UITableViewDataSource根据屏幕的尺寸，单元格的大小，为表格生成了五个单元格，并给每个单元格添加一个标签，用来显示数据源里的数字。

3、由于这五个单元格都可以在屏幕上看到，所以它们的状态都是使用中。

4、当用户使用手指向上滑动表格时，红色的单元格从顶部滑出屏幕。

5、当红色的单元格滑出屏幕时，它的状态变为未使用。

6、由于手指是向上方滑动的，所以需要在紫色单元格的下方补充一个单元格。

7、这时通过表格视图的dequeueReusableCell方法，从单元格列表里，获取状态为未使用的单元格，并将它放置在紫色单元格的下方。

8、同时更改单元格里的标签内容为6，并将它的状态设置为使用中。

9、红色单元格和其它的单元格，继续随着手指向上方滑动。

10、橙色单元格从顶部滑出屏幕，它的状态变为未使用。接着再次重复第4到9的步骤。

11、这样随着手指的滑动，不停的将从顶部滑出屏幕的单元格，移到屏幕的底部，并更新它的内容，从而实现对单元格的复用，以有限的单元格，显示无限的内容！

注：App Store下载《Xcode互动教程app for iOS开发》，可以在app中查看动画效果。

使用AVAudioPlayer播放音频的特点

AVAudioPlayer可以播放任意长度的音频文件、支持循环播放

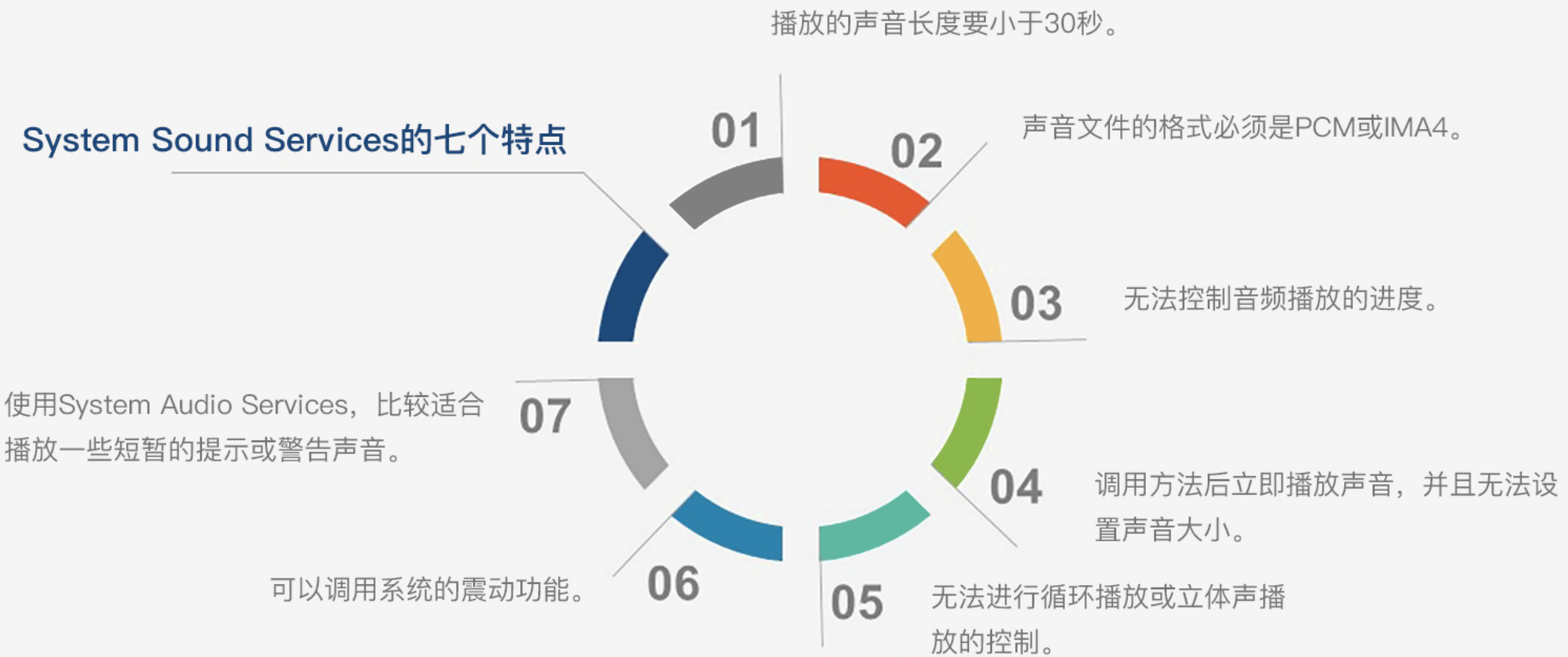
1、AVAudioPlayer有一个很强大的功能，可以很方便地调节左右声道的音量，从而实现很酷的立体声效果。因此AVAudioPlayer很适用于游戏中的音频播放。

2、由于AVAudioPlayer没有队列这个概念，因此它只能播放一个指定路径的音频。如果需要播放多个音频，可以通过创建多个AVAudioPlayer实例来实现。

3、AVAudioPlayer的其它特点如右图所示：

使用System Sound Services播放音频的特点

System Sound Services是最底层，也是最简单的音频播放服务



iOS开发中的音频的播放技术

在iOS中通常分别使用System Sound Services和AVAudioPlayer来完成音效和音乐的播放。
此处还可以选择Audio Queue Services和Open AL技术，这4种音频播放技术各有自身的特点：

System Sound Services

System Sound Services是最底层，也是最简单的音频播放服务，通过提供的C语言接口，允许开发者通过简单地调用AudioServicesPlaySystemSound方法，即可播放一些简短的音频文件。

AVAudioPlayer

AVAudioPlayer可以播放任意长度的音频文件、支持循环播放、可以同步播放多个音频文件、控制播放进度以及从音频文件的任意一点开始播放等。

Audio Queue Services

可以完全实现对声音的控制。开发者可以将声音数据从文件中读取到内存缓冲区，并对声音数据进行特殊处理，比如进行声音的快速、慢速播放，或者改变声音的音色等。

Open AL

和Open GL类似，Open AL也是一个跨平台的开源音频处理接口，它为音频播放提供了一套更加底层、更加精细的方案，特别适合具有复杂音频使用场景的游戏开发。

同步请求和异步请求

网络请求中的同步请求和异步请求的概念

1、同步意为着线程阻塞，以同步的方式从网络请求数据，一旦发送同步请求，应用程序将停止用户交互，直至服务器返回数据完成，才可以进行下一步操作。

2、也就是说同步就意味着阻塞线程，在同步请求过程中，应用程序的主线程不响应其它事件直到同步请求结束。

3、和同步请求相比，异步请求不会阻塞主线程，而会建立一个新的线程来操作，用户发出异步请求后，依然可以进行其它的操作，应用程序程序可以继续运行。

4、也就是说异步请求不会阻塞主线程对其它事件的响应，所以用户体验会优于同步请求。

GET和POST请求

每个GET和POST都由一系列HTTP请求头组成

Get请求

1. GET使用MIME类型的urlencoded文本的格式传递参数。
2. GET是从服务器上获取数据。
3. GET方式通过URL提交数据，数据在URL中可以看到。
4. GET方式提交的数据，最多只有1024字节。
5. 如果需要提交的是非敏感数据，那么可以使用操作方便、简单的GET方式。

Post请求

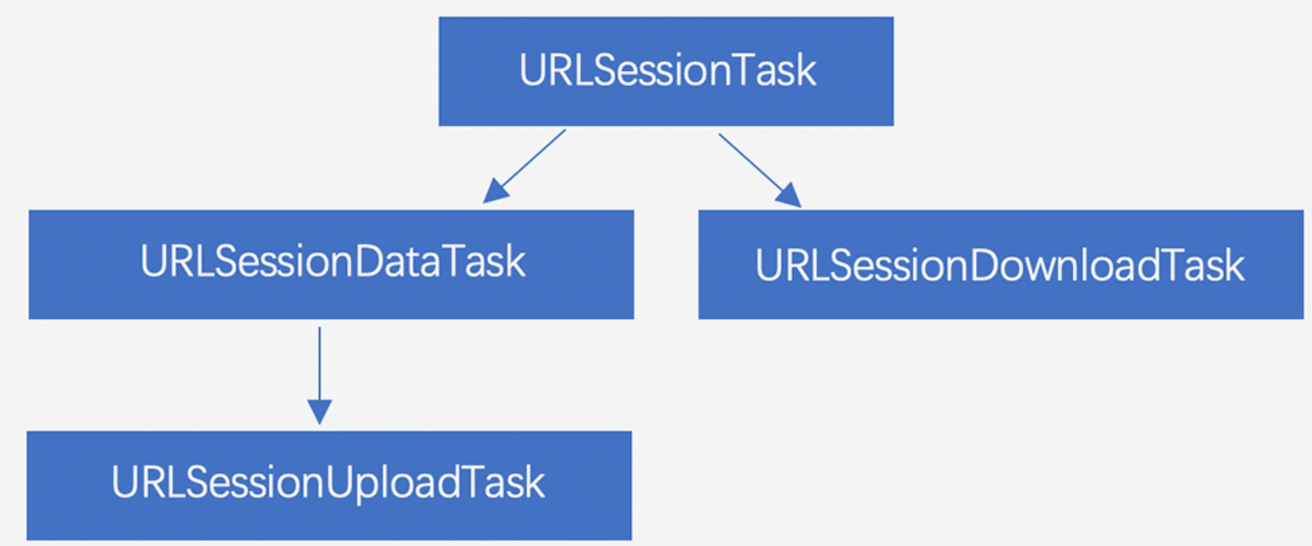
1. POST的参数是放在实际的HTTP请求消息内部被传送。
2. POST是向服务器传送数据。
3. POST方式是将数据放置在HEADER内提交。
4. POST方式没有数据量方面的限制。
5. 如果用户输入的数据包含敏感数据，那么还是推荐使用 POST为好。

使用URLSession进行网络请求

在2013的WWDC，苹果推出了URLSession作为NSURLConnection的替代

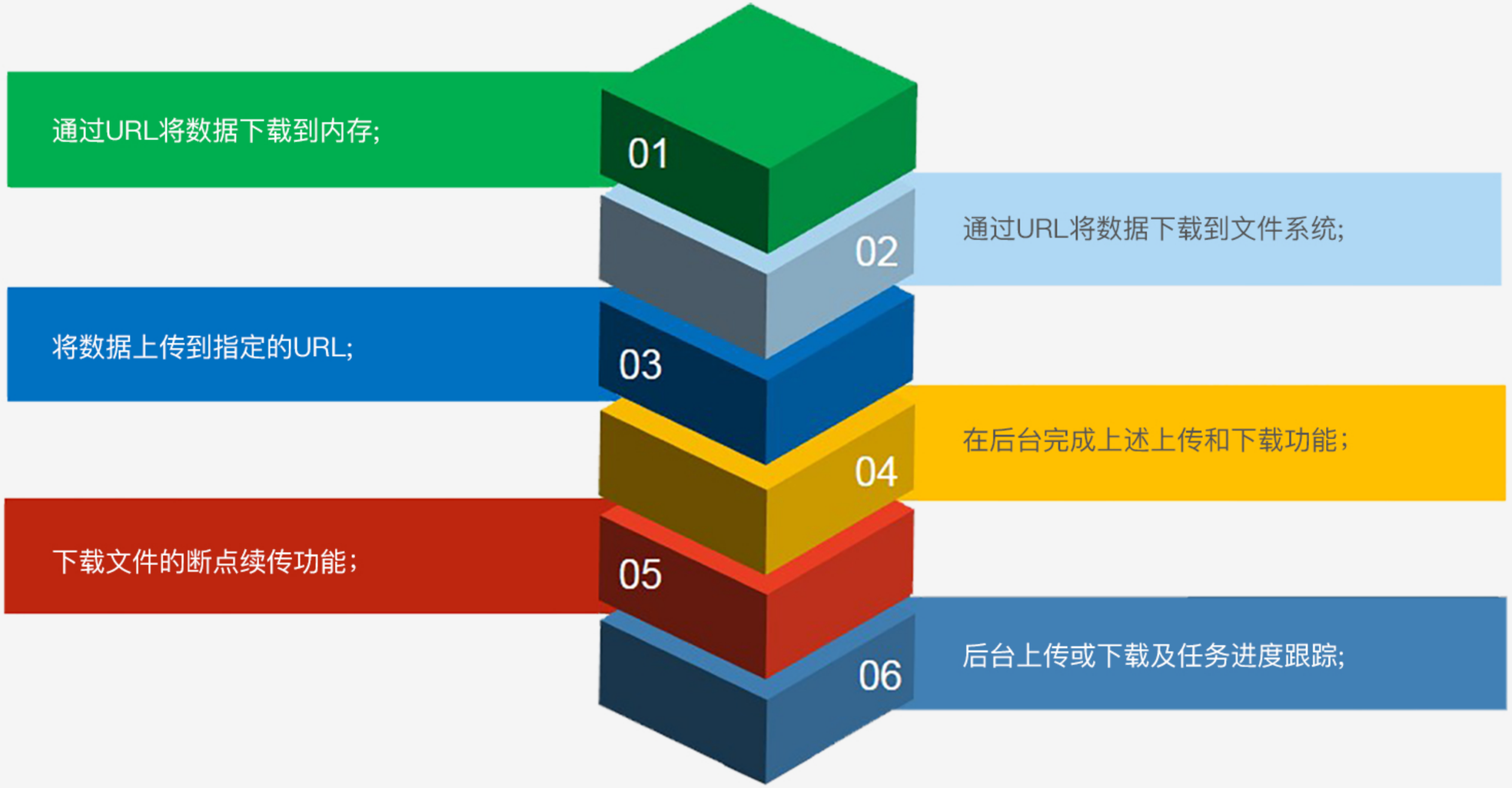
1、URLSession是一组相互依赖的类，它的大部分组件和NSURLConnection中的组件相同。如URLRequest、URLCache等。

2、而URLSession的不同之处在于，它将URLConnection替换为URLSession和URLSessionConfiguration，以及3个URLSessionTask的子类：URL-SessionDataTask, URLSessionUploadTask和URLSessionDownloadTask。



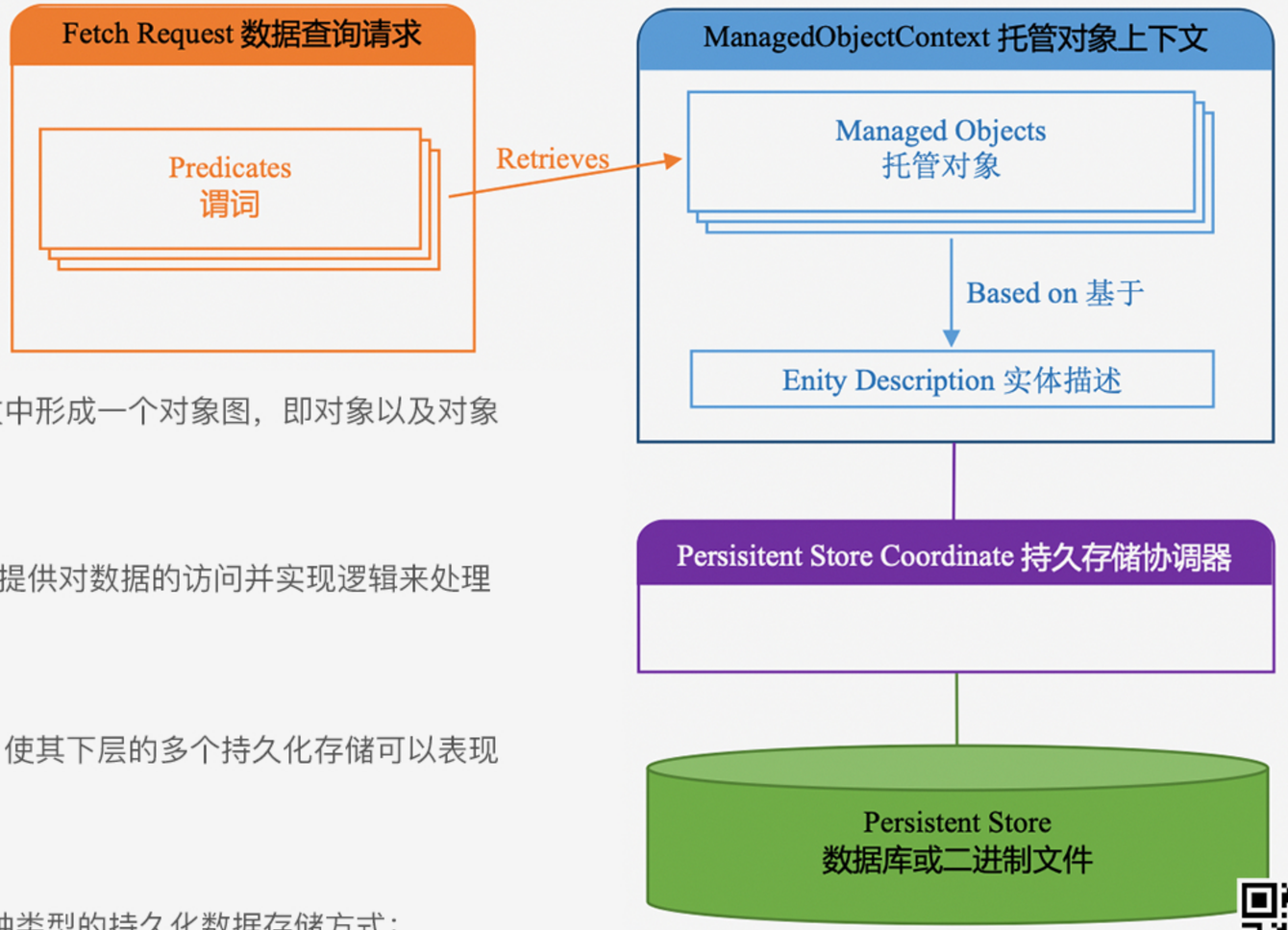
使用URLSession进行网络请求

URLSession类支持三种类型的任务：请求数据，下载和上传。



Core Data框架中各种对象之间的关系

Core Data用于存储和管理应用程序中，MVC设计模式里的模型层的数据。



Fetch Request数据查询请求：

使用托管对象上下文检索数据时，会创建一个获取请求fetch request。最简单的获取请求必须指定一个实体的名称，也可以包含一个谓词对象，通过谓词设置查询对象必须符合的查询条件。

ManagedObjectContext托管对象上下文：

当从持久化存储中获取托管对象时，这些对象的临时副本会在上下文中形成一个对象图，即对象以及对象之间的联系，然后便可以任意修改这些对象了。

Managed Objects托管对象：

Core Data框架中的托管对象是一种含有应用程序数据的对象类型，提供对数据的访问并实现逻辑来处理数据。所有托管对象都必须通过ManagedObjectContext进行注册。

Persistent Store Coordinate持久化存储协调器：

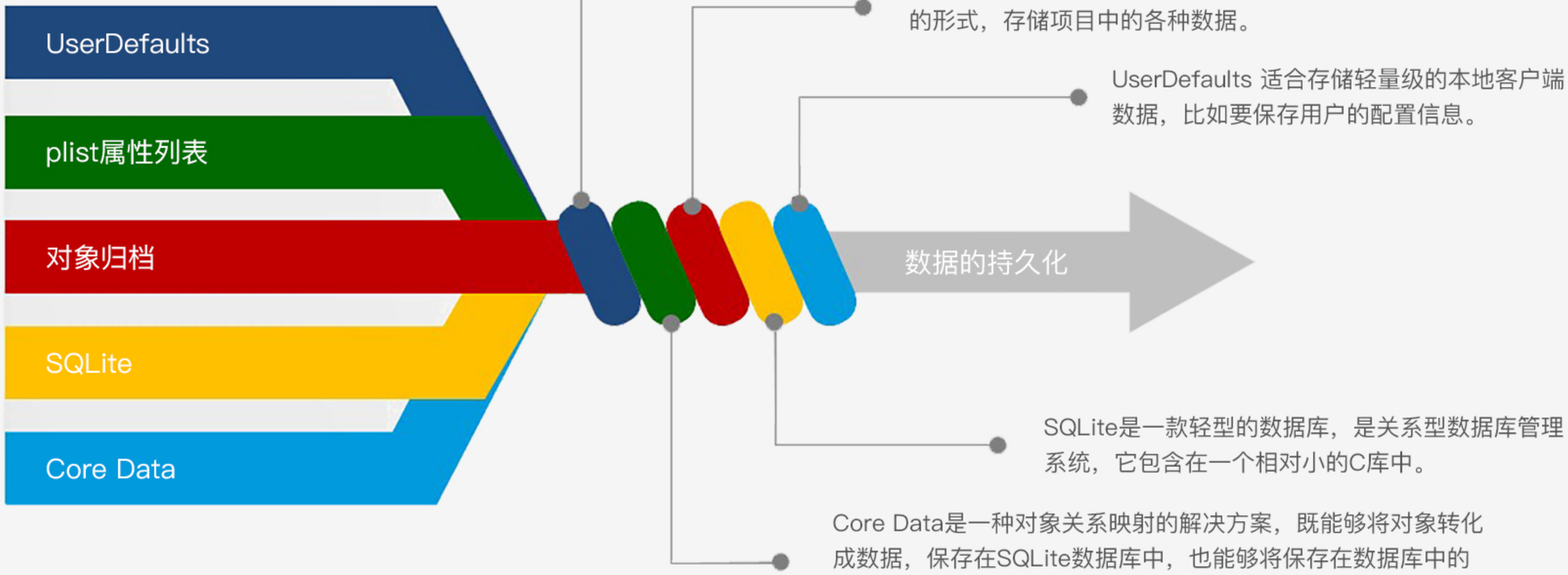
持久化存储协调器为一个或多个托管对象上下文提供一个访问接口，使其下层的多个持久化存储可以表现为一个聚合存储。

Persistent Store持久化存储：

负责数据和托管对象上下文中的对象之间的映射。Core Data支持4种类型的持久化数据存储方式：SQLiteStore、XMLStore、BinaryStore、InMemoryStore。

iOS中的数据持久化方式

一个功能丰富的应用程序，它的数据不仅存储在内存中，而是将更多的数据写入磁盘，进行持久化存储。



UserDefaults 适合存储轻量级的本地客户端数据，比如要保存用户的配置信息。

plist文件为属性列表文件，以类似于键值对（key-value）的形式，存储项目中的各种数据。

UserDefaults 适合存储轻量级的本地客户端数据，比如要保存用户的配置信息。

SQLite是一款轻型的数据库，是关系型数据库管理系统，它包含在一个相对小的C库中。

Core Data是一种对象关系映射的解决方案，既能够将对象转化成数据，保存在SQLite数据库中，也能够将保存在数据库中的数据还原成对象。

iOS开发中的三项主流多线程技术

这三种技术各有特点，开发者可以根据自己项目的情况，选择使用其中的一种技术。

1、Thread是三种多线程技术里面相对轻量级的，也是使用起来最需要开发者负责的。开发者需要自己管理线程的生命周期，以及线程之间的同步操作。

2、使用Thread创建的线程，拥有对数据相同的访问权限。

3、因此需要协调多个线程对同一数据的访问，通常的做法是在访问之前加锁。

4、使用Operation可以很方便的管理线程的并发数量。

5、Operation可以封装操作，然后将创建好的Operation对象放到OperationQueue队列中，队列便开始启动新线程去执行队列中的操作。

6、和Thread相比,Operation的优点是不需要关心线程管理、数据同步的事情，可以把精力放在业务逻辑上。

7、Grand Central Dispatch（GCD）很大程度上是和block配合使用的，所以GCD更加简单和易用。

8、GCD是为多核的并行运算提出的解决方案，所以会自动利用更多的CPU内核（比如双核、四核），并且会自动管理线程的生命周期：创建线程、调度任务和销毁线程。

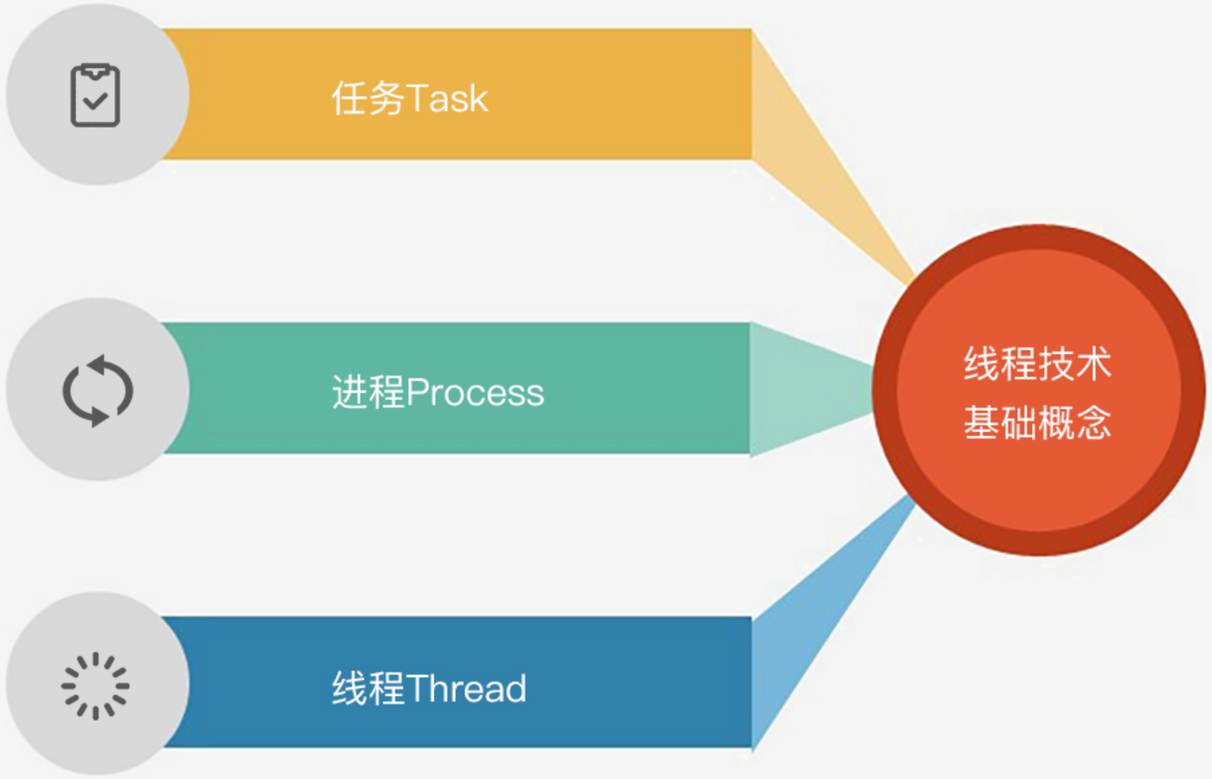
9、GCD被实现得如此轻量和优雅，使得它在很多场合，比创建消耗资源的线程更加实用和快捷！

并发、串行、同步和异步四个多线程术语

并发和串行表示任务的执行方式，而同步和异步表示要不要为任务开启新的线程。

任务、进程和线程

任务、进程和线程是三个不同的概念，而这三者之间也是具有一定相似性的，它们的具体区别如下所示：



1、任务是指由应用程序完成的一个活动。一个任务既可以是一个进程，也可以是一个线程。

2、任务是为达到某一共同目的的操作集合。

3、进程是具有一定独立功能的应用程序，是系统进行资源分配和调度的一个独立单位。

4、你可以把一个进程看成一个独立的程序，在内存中有其完备的数据空间和代码空间。

5、线程是某一进程里的一个单独运行的程序，是CPU调度和分派的基本单位。

6、一个进程由一个或多个线程构成，各线程共享相同的代码和数据，但各有其自己的堆栈。

线程的生命周期

在线程自创建至消亡的整个生命周期中包含多个状态

创建状态

对线程对象进行初始化。

就绪状态

通过调用线程对象的start方法，将线程对象添加到可调试线程池，等待CPU的调度。

运行状态

通过CPU的调度使线程处于执行状态，在结束前，线程的状态可能会在就绪和运行状态之间来回切换。

阻塞状态

调用Thread.sleep(_)，可以使线程在指定的时间范围内处于休眠的阻塞状态。

消亡状态

当线程执行完毕后将自动处于正常消亡状态。

线程的Stack space和Priority

Stack space and thread priority

线程的Stack space

1、每创建一个新的线程，都需要一些内存和消耗一定的CPU时间。

2、在mac OS中，主线程的栈空间大小为8 MB，而在iOS中，主线程的栈空间大小为1 MB，并且是不可修改的。

3、应用程序子线程的默认栈空间大小为512 KB，栈空间不是立即被创建分配的，它会在线程的使用过程中逐渐增加。

4、子线程允许分配的最小栈空间是16KB，并且必须为4KB的整数倍。

线程的优先级

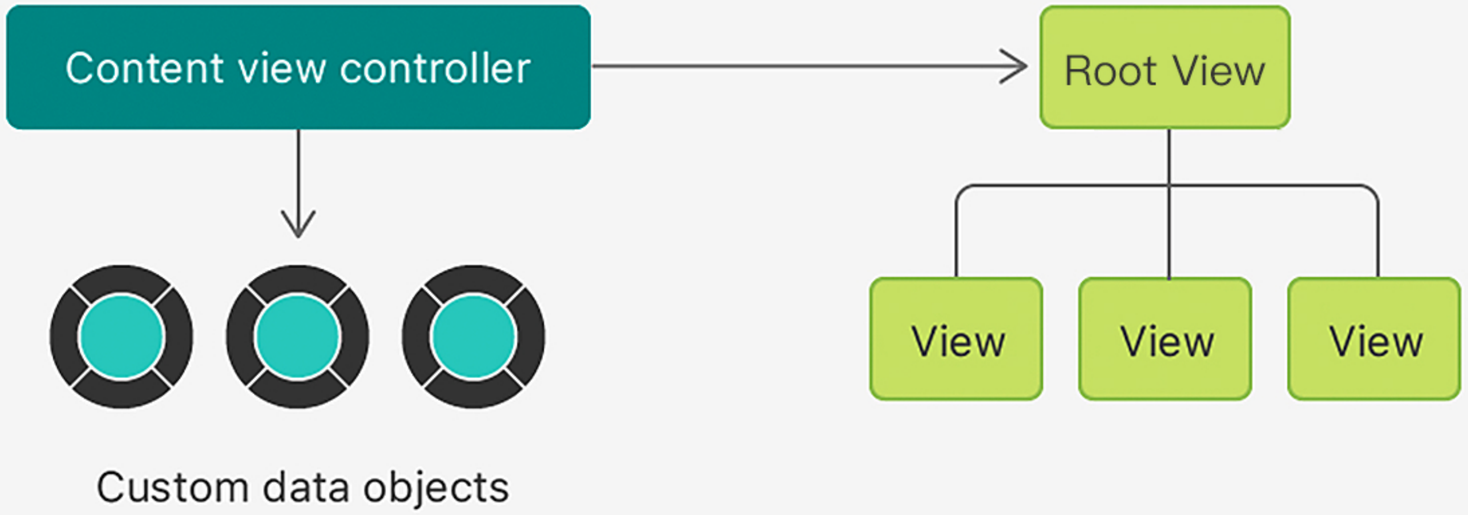
5、线程的优先级属性threadPriority是一个0.0～1.0之间的浮点数，1.0表示最高的优先级，这和iOS操作系统的优先级是一致的，线程默认的优先级是0.5。

6、在执行顺序方面，优先级较高的线程先执行的可能性高于优先级较低的线程。

7、但是优先级较高的线程并不是100%的比优先级较低的线程先执行，只是优先级高的线程得到CPU调度的机率更高。

使用Main run loop处理与用户相关的事件

调整视图层级顺序的四种方式



1、在模型–视图–控制器设计范例中，视图控制器既要处理在屏幕上显示信息的视图对象，还要处理存储应用内容的数据对象。

2、具体而言，视图控制器管理视图的层次结构，同时根据数据的变化，及时更新视图的显示状态。

3、每个 UIKit 应用都严重依赖视图控制器来呈现内容，并且您经常需要自定义一些视图控制器，来管理视图和 UI 相关逻辑。

参考资料：https://developer.apple.com/documentation/uikit/view_controllers/displaying_and_managing_views_with_a_view_controller

线程和Run loop

每条线程都有唯一的RunLoop对象与之对应

一、Run loop就像它的名字一样，是线程里的对收到的事件进行处理的一个循环。

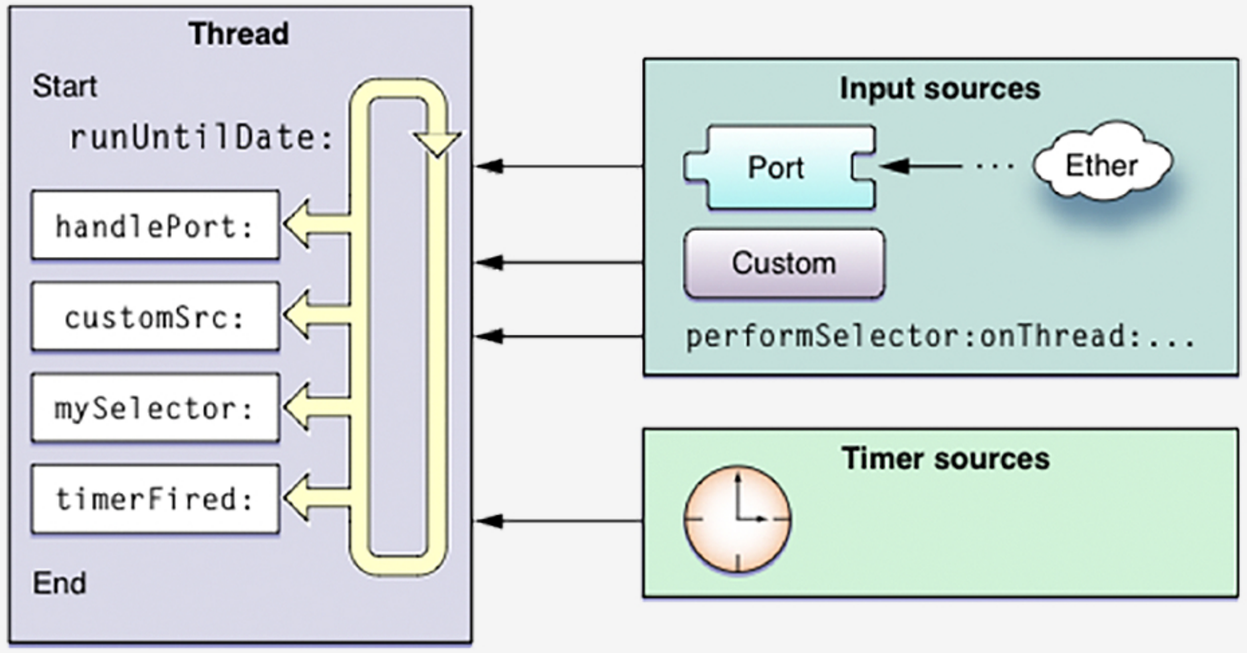
二、每条线程都有唯一的RunLoop对象与之对应，主线程的RunLoop是自动创建并启动的。

三、Run loop是通过代码的while循环来驱动的。在循环中使用RunLoop对象接收事件，并调用相关函数。

四、Run loop可以从两个不同的事件源中接收消息：

1、Input sources投递异步消息：使用port和自定义的输入源与其他线程进行通信，或者使用perform函数：

self.perform(#selector(controller.threadAction), with: nil，afterDelay: 2)



2、Timer sources：在计划的时间或重复的时间间隔内传递同步消息：

Timer.scheduledTimer(timeInterval: 1, target: self, selector: #selector (controller.timerAction(_:)),
userInfo: nil, repeats: true)

参考资料：https://developer.apple.com/documentation/foundation/runloop

字符和字形

Characters and Glyphs

1、字符是一种抽象的概念，它是使用语言进行书写时的最小单位，例如英文字母、阿拉伯数字或者单个的中文象形文字。

2、文字是用来沟通的，所以字符需要通过字形来展示自己，只不过一个字符可以根据不同的场景，使用不同的字形进行展示。如图所示，同一个“果”字，可能拥有不同的尺寸和不同的外观，可以是粗体也可以是斜体。

3、有时一个字符可以通过两个字形来表示。例如法文字符[é]，是通过字形[e]和一个声标字形[´]相结合组成的。

4、而有些情况一个字形有可能同时展示多个字符，例如在英文书写中经常出现的连字情况。在书写文字时基于美观方面的考虑，有时会使两个字符的距离尽可能地靠近，从而产生字符之间的连字现象，如右图所示。在相同的字体Candara下，左侧是三个字符ffh在未连字和标准连字下的状态，右侧是两个字符fi在未连字和标准连字下的状态，处于连字情况下的若干个字形，被结合为一个字符。

字型与字体

Typefaces and Fonts

1、字型是一组在视觉上具有相似性的字体形状。例如由Stanley Morrison在1931年为伦敦时报设计的Times字型，在该字型下的英文字母都具有相似的视觉特性。对于我们开发者来说，可以简单地将Typefaces理解为Font family字体组。

2、字体Fonts是一组具有相同风格字型Typefaces的字形Glyphs，是文字的外衣。字体均由人工设计，即由字体设计师设计而成。

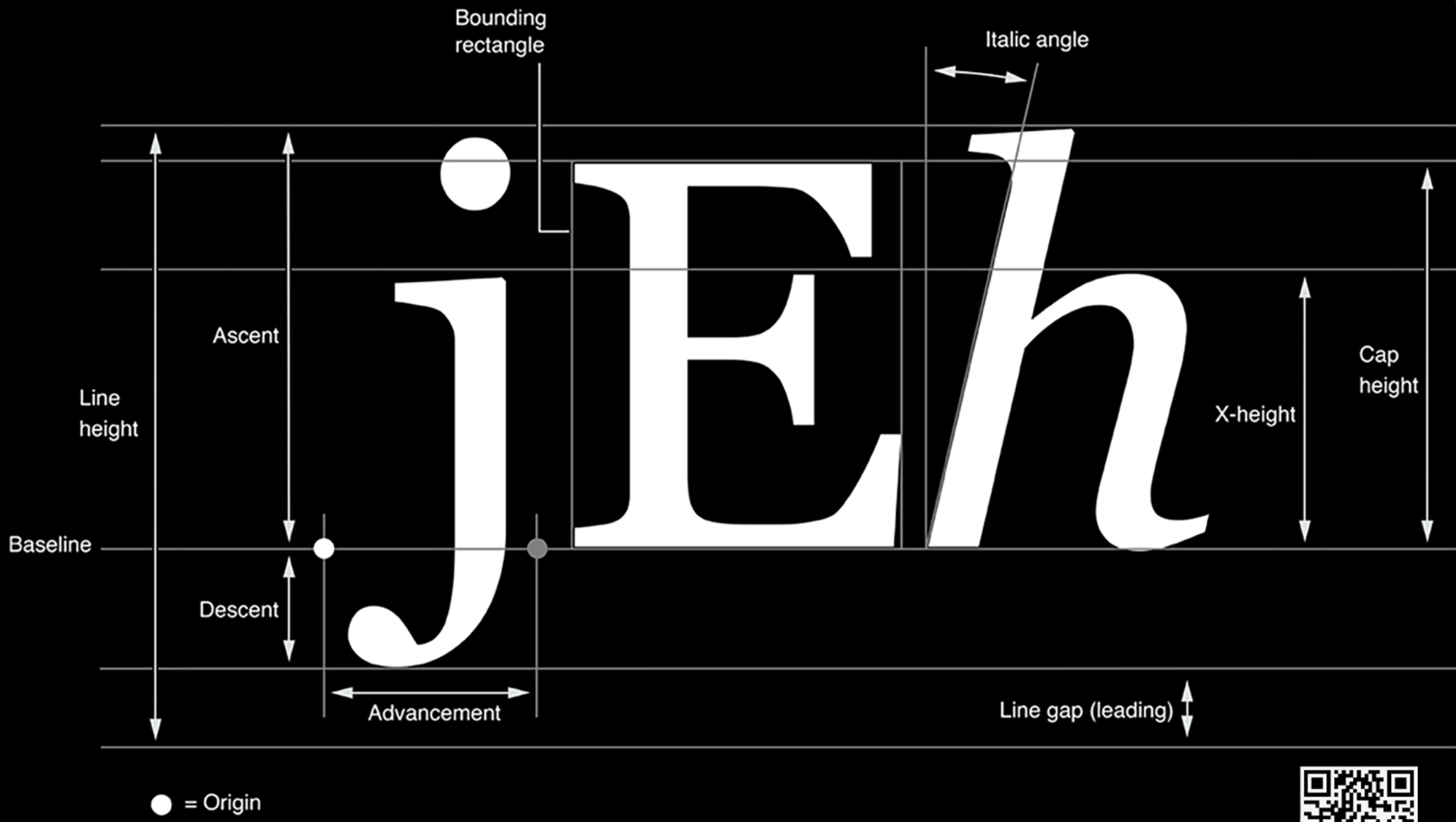
3、在图所示，分别是文字“Typefaces and Fonts”在Lucida字体组（字型）中四种字体的不同表示。四种字体按照从上往下和从左至右的顺序，依次是Lucida Black letter、Lucida Bright、Lucida Handwriting和Lucida Calligraphy。

文字的布局

Text Layout

1、文字的布局就是将众多字形Glyphs，通过一定的规则排列在显示设备上。

2、在Core Text框架中，字形的排列是基于一条不可见的线进行排列的，这条抽象的线被称为基线Baseline，如图所示。



字体布局概念列表

有关字形排列的名词，它们的含义如表所示

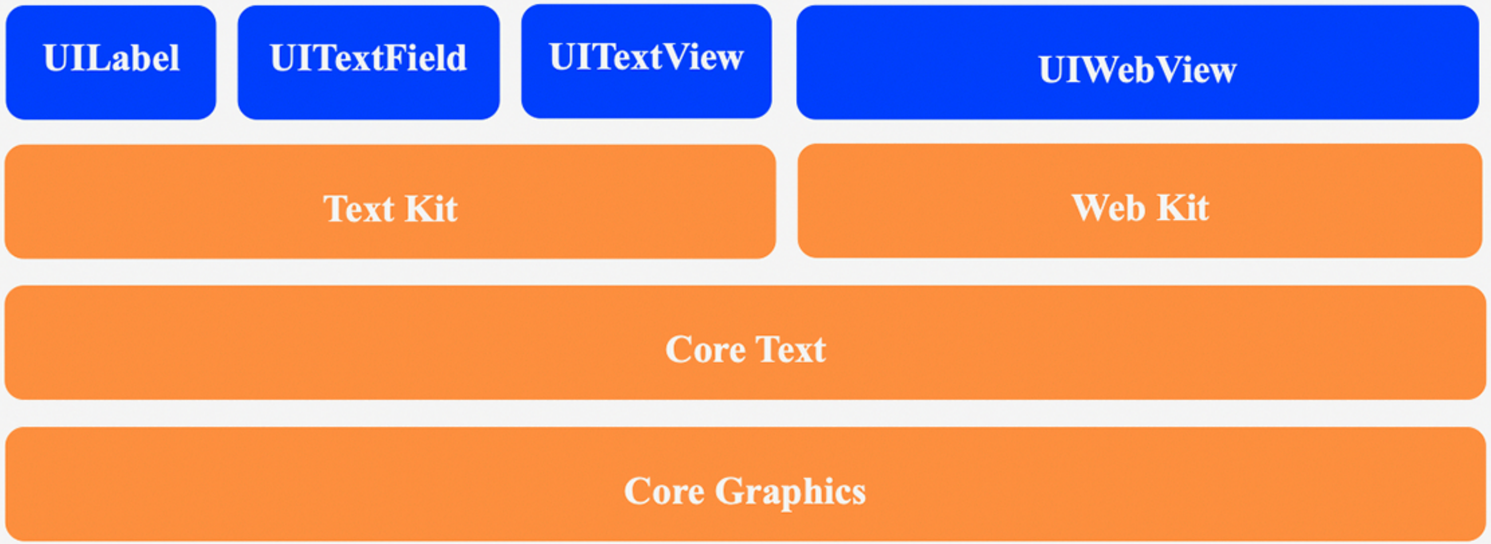
名称	功能说明
Origin	表示位于基线上的、一个字形在排列时基于的原点
Baseline	表示字形在排列时，字形底部紧靠着的一条直线
Line height	表示一行字形的最大高度，等于 Ascent 和 Descent（取其绝对值）及 Line gap（leading）三者之和
Ascent	表示上行高度，是基线与字形最高点之间的距离
Descent	表示下行高度，是基线与字形最低点之间的距离
Line gap(leading)	表示行距，是上方一行的最低点与下方一行的最高点的距离
Advancement	表示前后两个字形的原点之间的距离
Bounding rectangle	能够容纳字形的最小矩形框
Italic angle	斜体字形在垂直方向上沿顺时针的倾斜角度
X-height	基线至非突出的小写字母（如 a、x、e 等）最高点的距离
Cap height	基线至大写字母最高点的距离

Text Kit的使用

有关字形排列的名词，它们的含义如表所示

1、为了给开发者提供更高效的文字排版工具，Apple向开发者推出了Text Kit类库。该类库是建立在Core Text基础之上的，并且和UILabel、UITextView等控件紧密结合，在使用这些控件时可以直接使用Text Kit相关的功能。

2、Text Kit与其他类的关系如图所示，从示意图中可以得知Text Kit和Web Kit一样，都是建立在Core Graphics和Core Text框架之上。



Text Kit中的三个主要的类

灵活使用Text Kit的三个主要的类，可以快速进行富文本视图的创建和设计

NSTextStorage类

1、UITextView拥有NSTextStorage类型的属性textStorage，该类继承自Core Text中的NSAttributedString类，用来存储文本视图控件中的文本信息和属性。

NSLayoutManager类

1、NSLayoutManager是一个管理类，通过该类可以将多个NSTextContainer区域进行连接，从而使NSTextStorage中的数据能够在多个视图分开显示，以达到对文章分栏显示的效果。

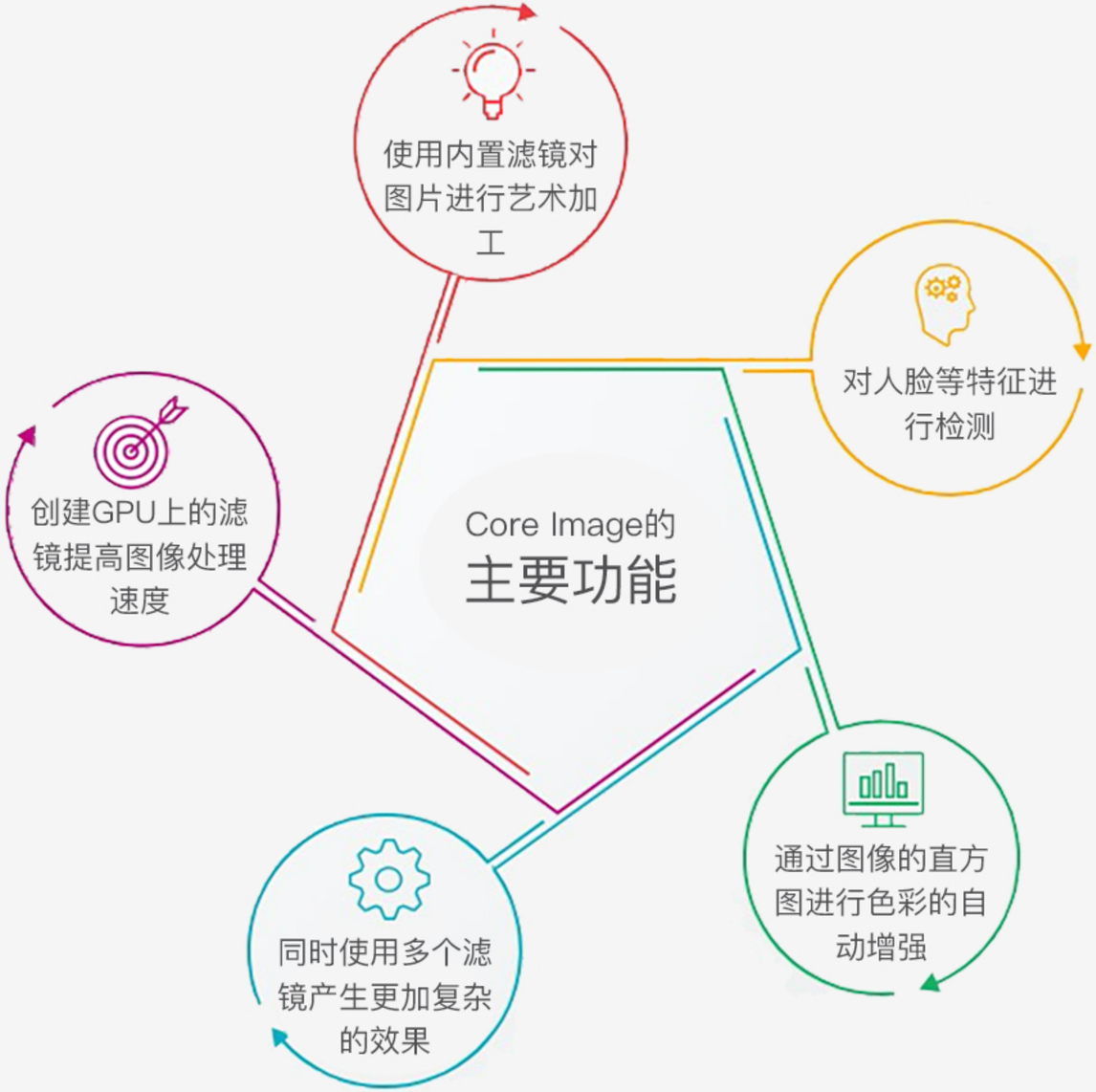
NSTextContainer类

1、每个UITextView都拥有NSTextContainer属性，该属性表示文本视图的显示区域。文本视t图中的所有文字都被填充在该区域。

2、NSTextContainer类有一个重要的属性exclusionPaths，通过该属性设置文字在排列时需要跳过的区域，从而实现图文紧密排列效果。

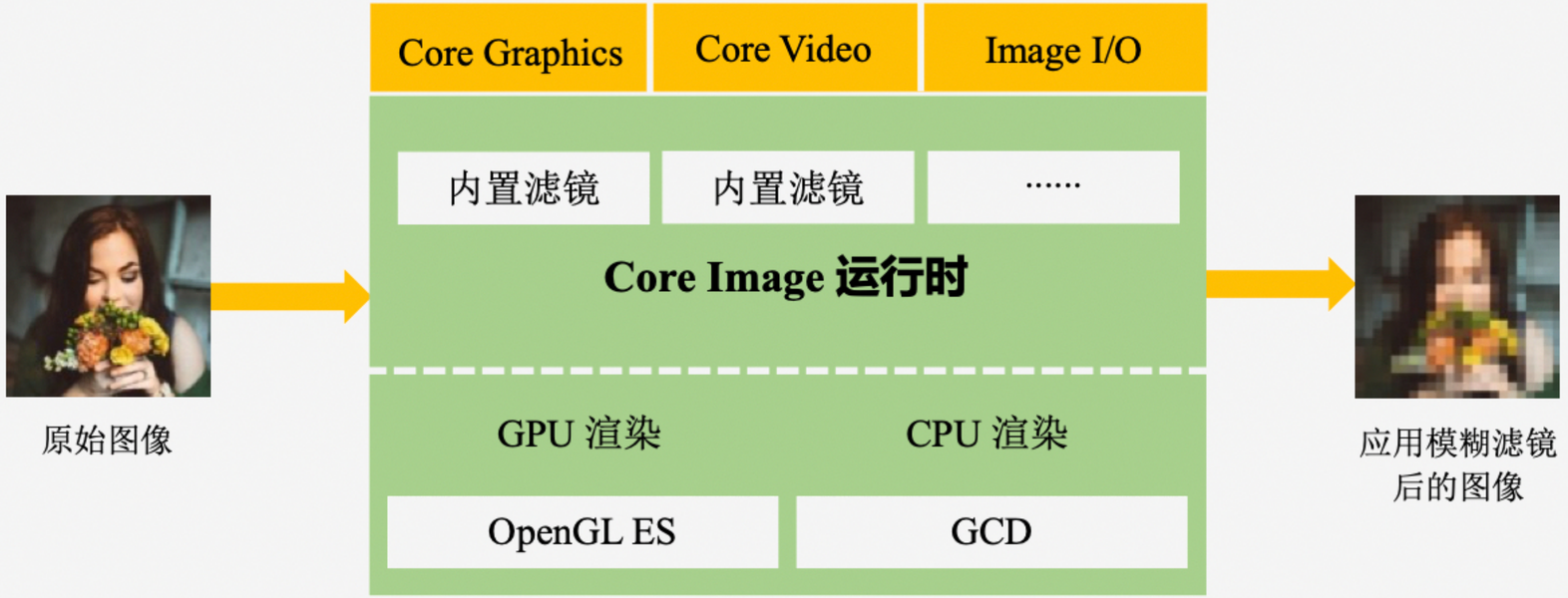
Core Image 图像处理框架

Core Image 提供了强大高效的图像处理功能



iOS系统中的Core Image框架工作图

使用Core Image框架对图片进行处理的流程



从工作图可以看出：

1、Core Image可以操作来自Core Graphics、Core Video和Image I/O框架的Image Data图片数据。

2、可以使用Core Image中的众多内置滤镜，对图片进行艺术加工。

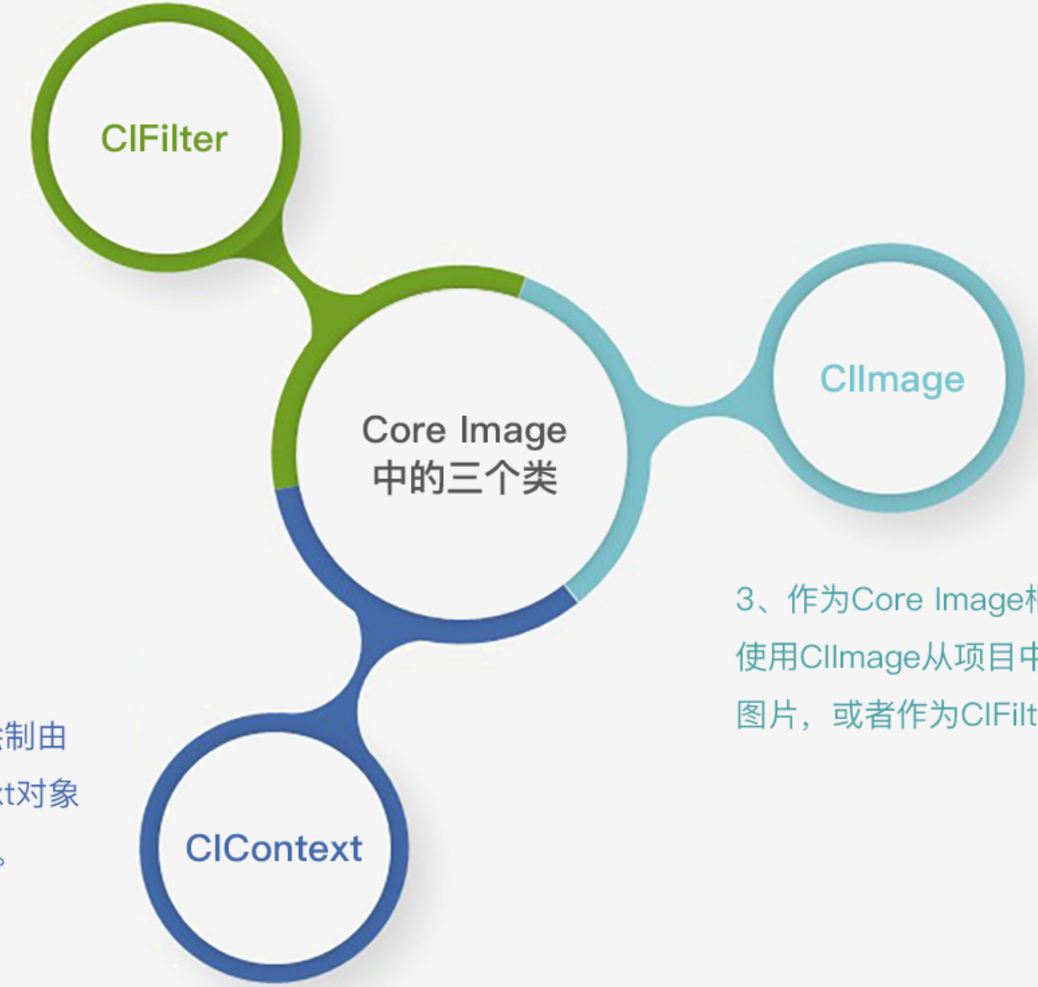
3、最后在GPU或CPU中进行处理结果的渲染并输出。

Core Image框架中的三个类

通过使用Core Image框架的三个类，就可以完成所有的图像处理工作

1、是进行图像处理的滤镜，每种滤镜都有不同的设置参数。

2、每个滤镜对象至少拥有一个输入参数，并且产生一个输出图像。



3、作为Core Image框架中的图像类型，可以使用CIImage从项目中读取一张图片作为输入图片，或者作为CIFilter对象的输入图片。

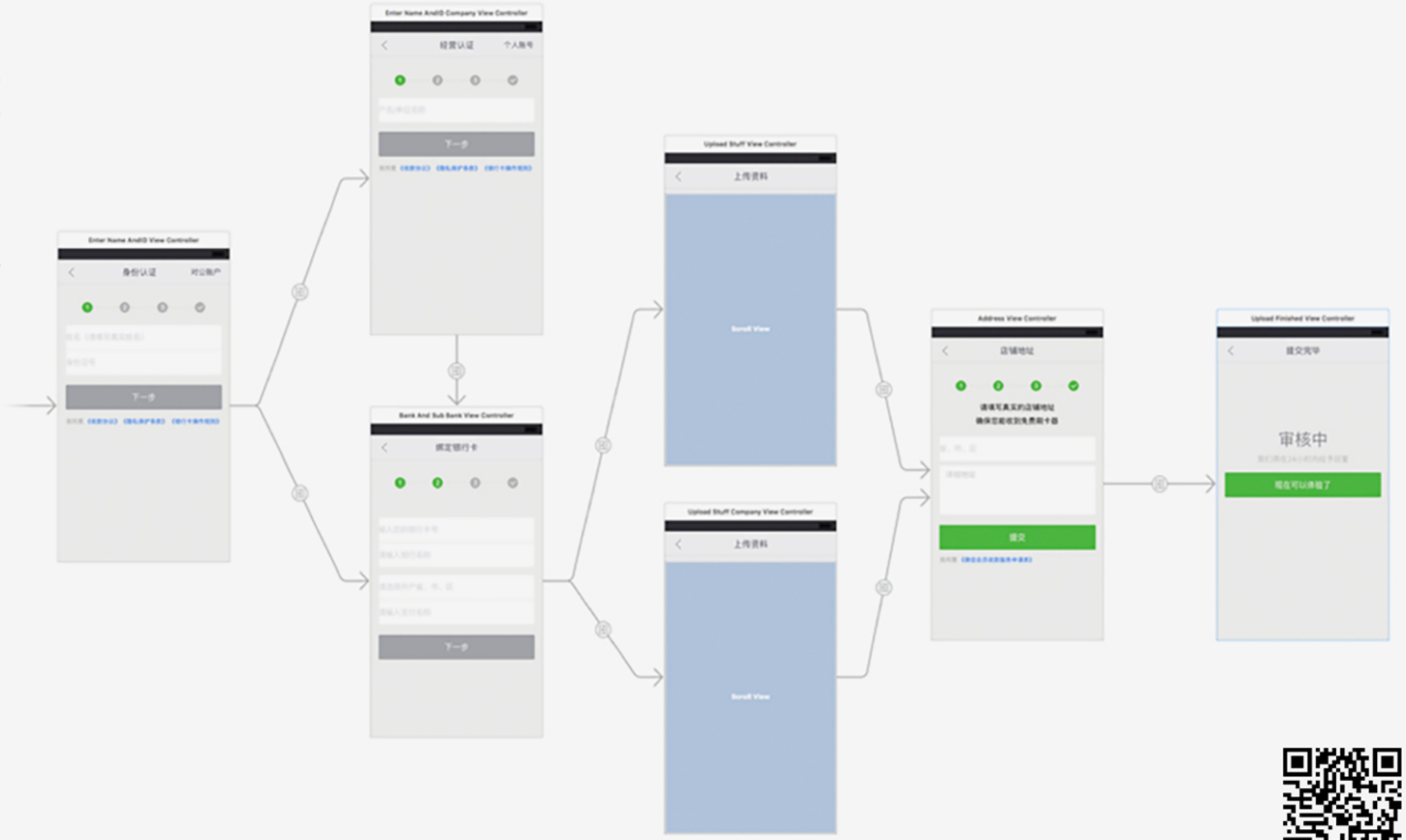
4、Core Image通过CIContext绘制由CIFilter处理后的结果，CIContext对象可以基于GPU，也可以基于CPU。

使用Storyboard绘制用户界面

开始使用故事板绘制用户界面

1、使用Storyboard故事板，可以在开发app界面时，极大地节省开发时间，提高开发的工作效率。

2、如图所示，在这个故事板中，建立了7个视图控制器，并设置了它们彼此之间的跳转关系，使得系统的业务流程更加清晰、明了。



Storyboard的优点和缺点

THE PROS AND CONS OF STORYBOARD

Storyboard的优点

1、通过简单地拖拽，往视图控制器添加各种界面控件，提高工作的效率，使开发工作更加人性化。

2、更好地查看和理解所有页面的外观，以及页面之间的跳转关系，新同事也能够快速融入到项目中。

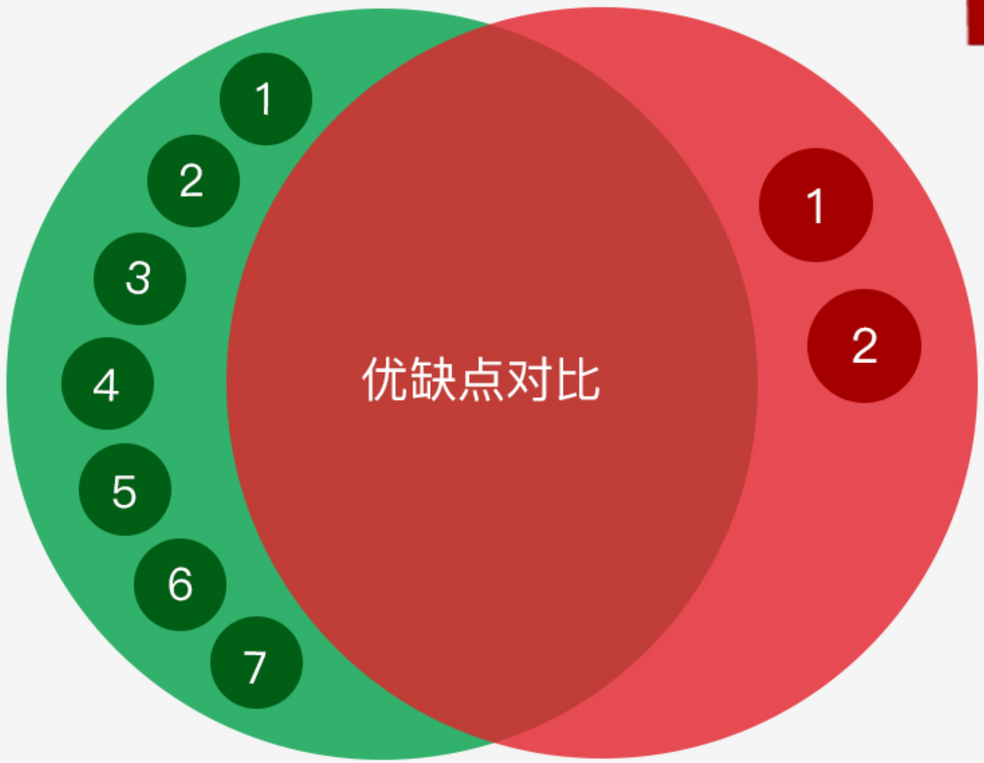
3、Storyboard快速实现了页面之间的跳转关系，从而节省大量的代码。

4、通过重写prepareForSegue方法，将视图控制器的所有跳转逻辑都汇聚一处，方便进行统一的界面跳转和参数传递。

5、Storyboard通过Cell ProtoType原型，以及Static静态Cell的特性，让表格实现起来更加容易。

6、通过将常见的功能比如注册、分享模块独立成一个Storyboard，可以实现功能的模块化和复用。

7、使用Storyboard可以通过拖拽建立约束关系，从而快速实现各种分辨率的适配。



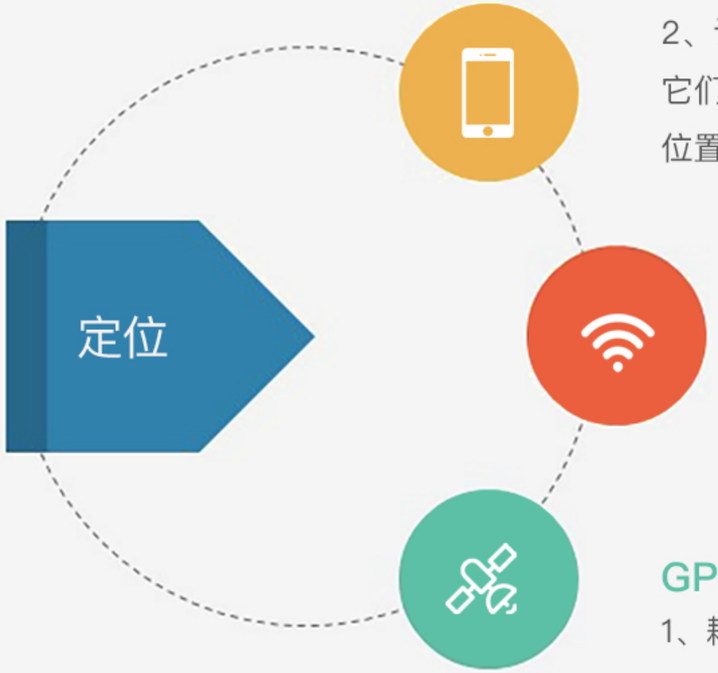
Storyboard的缺点

1、Storyboard的版本管理不太方便，对此你可以将Storyboard按照功能模块或开发人员进行拆分，每个开发人员负责编辑和提交各自的Storyboard。

2、当Storyboard包含越来越多的视图控制器时，打开Storyboard将越来越慢。不过这问题随着Xcode软件的升级已经被逐步优化。

iOS设备支持三种定位方式

三种定位方式各自的特点



手机基站定位

1、定位速度最快，并且耗电最小，只是误差范围比较大，通常在几百上千米左右。

2、该定位方式的原理是：每个手机基站都有一个标识符，iOS设备可以搜集周围所有收到信号的基站和它们的标识符，通过联网发送到苹果云端服务器，再由服务器根据这些基站的位置查询并计算出当前的位置，然后返回给手机。

WIFI定位

1、定位速度、耗电和精度三个指标都介于基站定位和GPS定位之间，精度在几十米左右。

2、其原理是iOS设备通过无线网卡周围的WIFI热点，获得它们的MAC地址，然后到苹果云端的服务器查询这个热点的位置，最后通过计算得到当前位置并返回给用户。

GPS定位

1、耗电最多，定位速度最慢，但是优点是定位的精度最高。

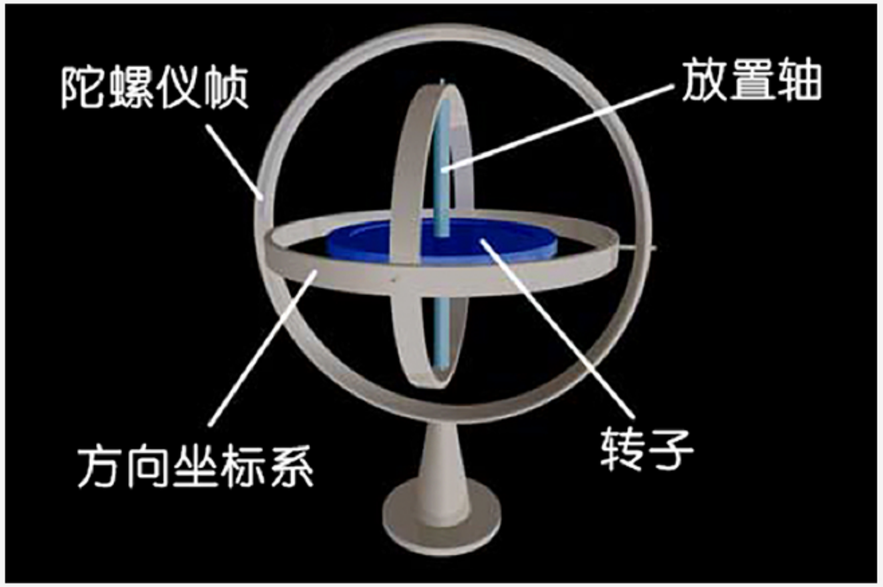
2、原理是利用卫星不断地广播信号，地面的GPS接收设备收到信号后，通过分析多个卫星信号，就可以计算出地球坐标，其定位精度可达10米以内的范围。

陀螺仪的原理和应用

Principles and applications of gyroscopes

1、陀螺仪的原理是：一个旋转物体的旋转轴所指的方向，在不受外力影响时是不会改变的。根据这个原理，制造出来陀螺仪，用它来保持方向。

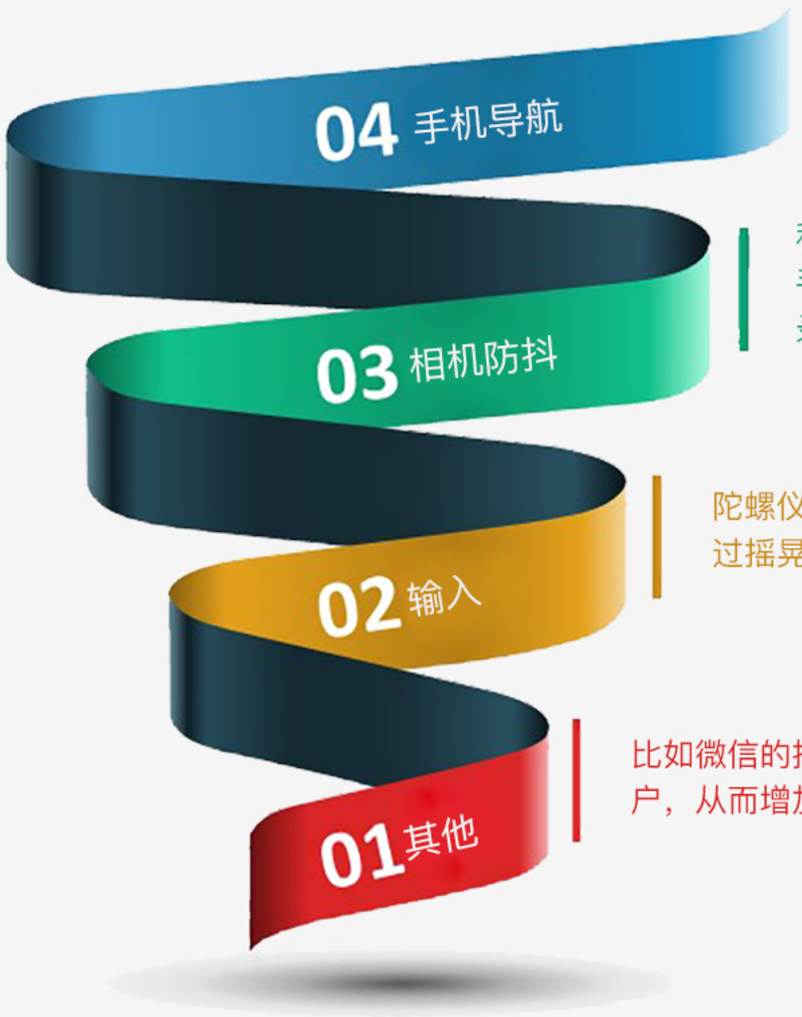
2、陀螺仪在工作时需要受到一个力，使它快速旋转起来，一般能达到每分钟几十万转，且可以工作很长时间。然后用多种方法读取轴所指示的方向，并自动将数据信号传递给控制系统。



陀螺仪结构图

陀螺仪的主要用途

THE MAIN USES OF GYROSCOPES



通过配合GPS设备，手机的导航能力达到前所未有的水准。iPhone手机在安装相应的软件后，其导航能力不亚于目前很多船舶、飞机上用的导航仪。

和iPhone和iPad上的摄像头配合使用，在按下快门时，记录手的抖动动作，将手的抖动反馈给图像处理器，可以抓拍到更加清晰稳定的图片。这使得拍照、录制视频的质量得到很大的提升。

陀螺仪相当于一个立体的鼠标，所以经常被用在赛车、战机类的游戏中，可通过摇晃设备来控制赛车和战机的方向。

比如微信的摇一摇功能，通过摇晃手机可以匹配到同一时段触发该功能的微信用户，从而增加用户间的互动和微信应用程序的黏度。

加速计的原理和应用

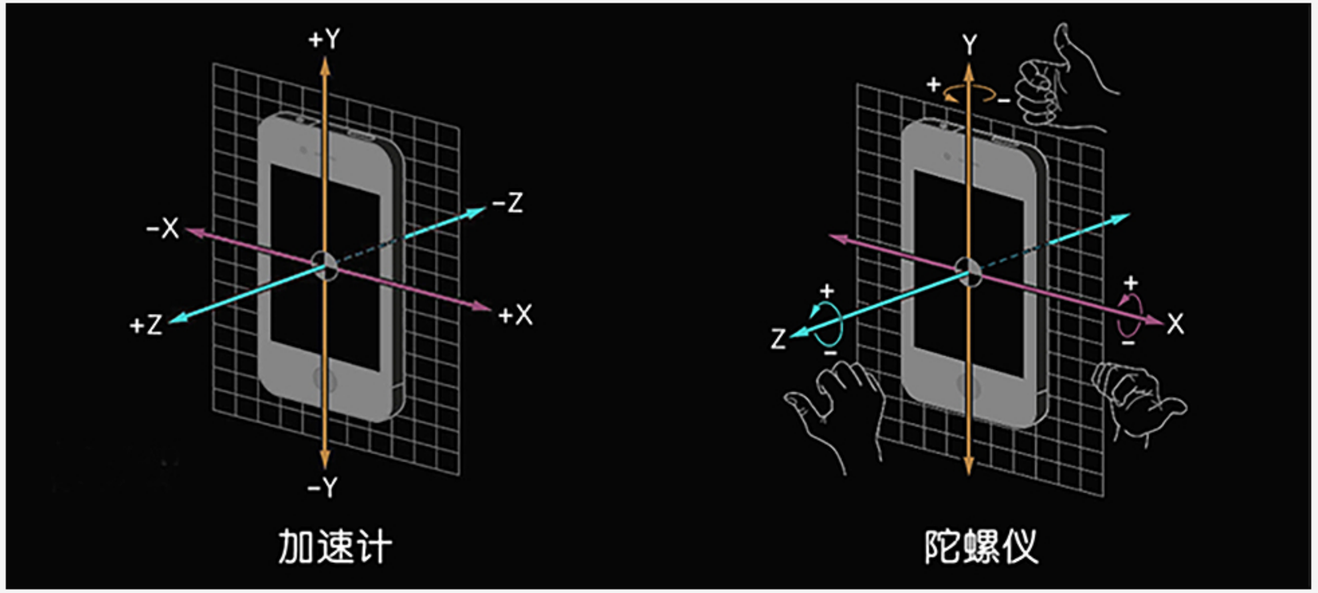
THE MAIN USES OF GYROSCOPES

1、iPhone 通过内置方向感应器来对动作做出反应，方向感应器的实现靠的是iPhone的内置加速计。

2、iPhone所采用的加速计是三轴加速计，分为X轴、Y轴和Z轴。这三个轴所构成的立体空间，足以侦测到你在iPhone上的各种动作。

3、在实际应用时通常是以这三个轴或其中的任意两个轴所构成的角度，来计算iPhone倾斜的角度，从而计算出重力加速度的值。

4、陀螺仪和加速计看起来很接近，不过两者之间是有区别的。加速计只能侦测物体的移动行为，并不具备精确侦测物体角度改变的能力，陀螺仪可以侦测物体水平改变的状态，但无法计算物体移动的激烈程度。



iOS应用程序的本地化

当你计划通过App Store发布应用时，也应当考虑国外市场的广大用户

应用名称的本地化简要步骤

1、创建InfoPlist.strings文件

2、对该文件进行本地化处理

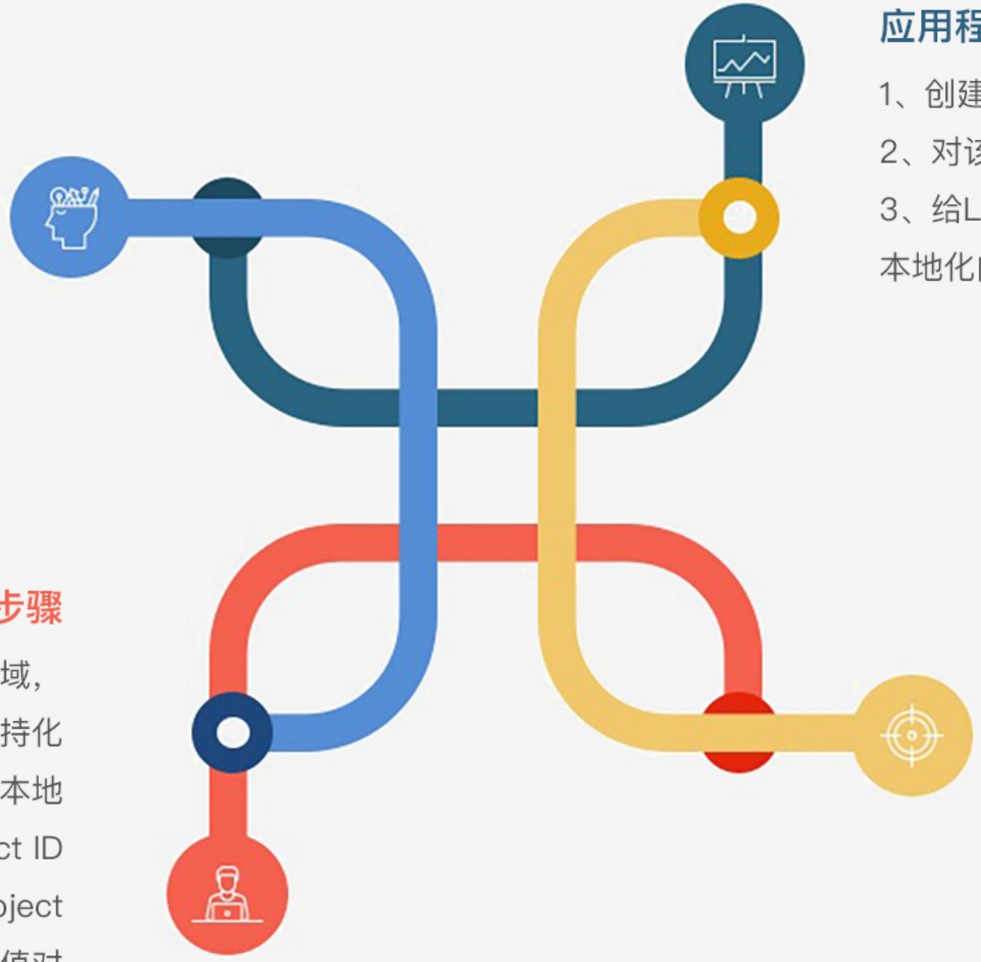
3、给InfoPlist.strings添加各语言版本的本地化支持

Storyboard的本地化简要步骤

1、在Info面板的[Localization]设置区域，给故事板添加本地化支持化

2、在Identity Inspector面板获得需要本地化控件的Object ID

3、在故事板的本地化文件中，输入Object ID和其对应的值作为键值对



应用程序文字内容的本地化简要步骤

1、创建Localizable.strings本地化配置文件

2、对该文件进行本地化处理

3、给Localizable.strings添加各语言版本的本地化内容

图片素材的本地化简要步骤

1、选择图片，然后打开右侧的[File Inspector]，在Localization设置区域点击Localize按钮给图片添加本地化支持。

2、在Info面板的[Localization]设置区域，给图片添加本地化支持化

3、此时在图片名称的下方会多出几个本地化的预留图片，将各语言版本的图片粘贴覆盖这些预留图片即可

注：具体操作步骤请参考我们提供的互动教程

五步开启iOS开发职业生涯

由浅入深 循序渐进

Swift语言入门实例互动教程

详细讲解Swift语言大部分知识点，内存管理、错误处理、链式编程、常见的算法、还包含一个完整的项目实例！

移动端： 扫描下方二维码

网页端： http://hdjc8.com/hdjc/iosSwift/

Objective-C应用开发互动教程

详细讲解OC语言的大部分知识点。内存管理、多媒体、数据解析和加密、Sqlite数据库、网络多线程、内购等实用功能也都有涉及！

移动端： 扫描下方二维码

网页端： http://hdjc8.com/hdjc/iosOc/

Xcode教程 for iOS开发

230节大容量互动教程，简单而有趣，是手把手互动式学习iOS开发的新工具。

移动端： 扫描上方二维码

网页端： http://hdjc8.com/hdjc/iosXcode/

iOS开发中的神兵利器

手把手学习iOS开发中的强大的第三方类库，详细讲解Github热门的开源项目。助您快速、优雅地解决iOS开发中棘手的业务需求！

移动端： 扫描上方二维码

网页端： http://hdjc8.com/hdjc/iosGithub/

SwiftUI从入门到实战

196节大容量课程：详细讲解SwiftUI的方方面面。15个超级精彩的实例：包含美食、理财、健身、教育、电子商务等各行各业的App实例！

移动端： 扫描下方二维码

网页端： http://hdjc8.com/hdjc/swiftUI/